QorIQ Processor Enablement Tool Introduction

AMF-NET-T1020

Greg Hemstreet

Manager, Processor Expert Software

August 2013

Promise

- **Before you leave today you will**
  - Understand why configuration tools will help you
  - Understand some other tools for optimizing QorIQ devices
  - Configuration Tools specifically:
    - Have undergone a basic walkthrough of the tools
    - Used actual configurations and modified them based on cust
      requests to configure:
      - RCW – pre-boot loader settings
      - BOOTROM – P1/P2 devices without RCW use this tool
      - DDR – memory controller settings
      - Device Trees – Linux® hardware device tree settings
      - Data Path graphs – configuring the DPAA
  - Know where to get more information



Agenda

- **Explore Configuration Tools available**
- **Explore Optimization Tools available**
- **Describe how people get the tools**
- **What is using QCS?**
  - **Review each tool**
  - **Pre-boot loader / RCW configuration**
  - **DDR configuration**
  - **Device Tree Editor**
  - **Data Path graphs and configuring the DPAA**
- **Summary**
- **Where to get more information…**
- **Walkthrough Labs – backup slides**
  - Lab 1: Installing QCS
  - Lab 2: Pre-boot loader / RCW Configuration
  - Lab 3: Bootrom generator
  - Lab 4: DDR configuration
  - Lab 5: Device Tree Editor
  - Lab 6: Data Path graphs and configuring the DPAA

QorIQ Software Enablement Strategy

- ✓ Provide Development Systems
  - Complete boards for evaluation of QorIQ devices
- ✓ Provide Runtime software for QorIQ products
  - MKV, Hypervisor, Linux BSPs, Reference Designs
- ✓ Provide Bring-up tools and development systems
  - GNU tools, CodeWarrior debuggers, probes, boards
- ✓ Provide Configuration Tools to support your application of QorIQ on your custom board.
  - RCW, BootROM, Pin Mux
  - DDR Configuration
  - Device Tree Editing
  - DPAA Configuration
- ✓ Provide Optimization Tools to support runtime visibility into these complex parts to help calibrate and debug your systems.
  - DDR validation tool to ensure DDR functionally configured for custom board
  - Scenarios tool for collecting and visualizing runtime trace data
  - Packet tool for understanding the flow of packets within a QorIQ device

freescale™

Processor Expert Software

QorIQ Configuration Suite





Why QorIQ Configuration Suite?

- Configuration of QorIQ processors is increasing in complexity
  - Even more complexity is around the corner
  - We support many, many configuration settings
- Reference manuals are huge and intimidating to new customers
- Configuration problems during board bring-up are HARD and COSTLY
- Learning command line tools requires more training, etc.

- **Solution/Strategy to solve these problems:**
  - **Extensible suite of tools with a common user interface**
  - Consolidate into a common tools framework (Processor Expert)
  - Provide new device support aligned with silicon roadmap
  - Add more configuration tools over time
  - Allow customers to add their own configuration tools to extend what we offer …

QorIQ Configuration Suite – Now Available!

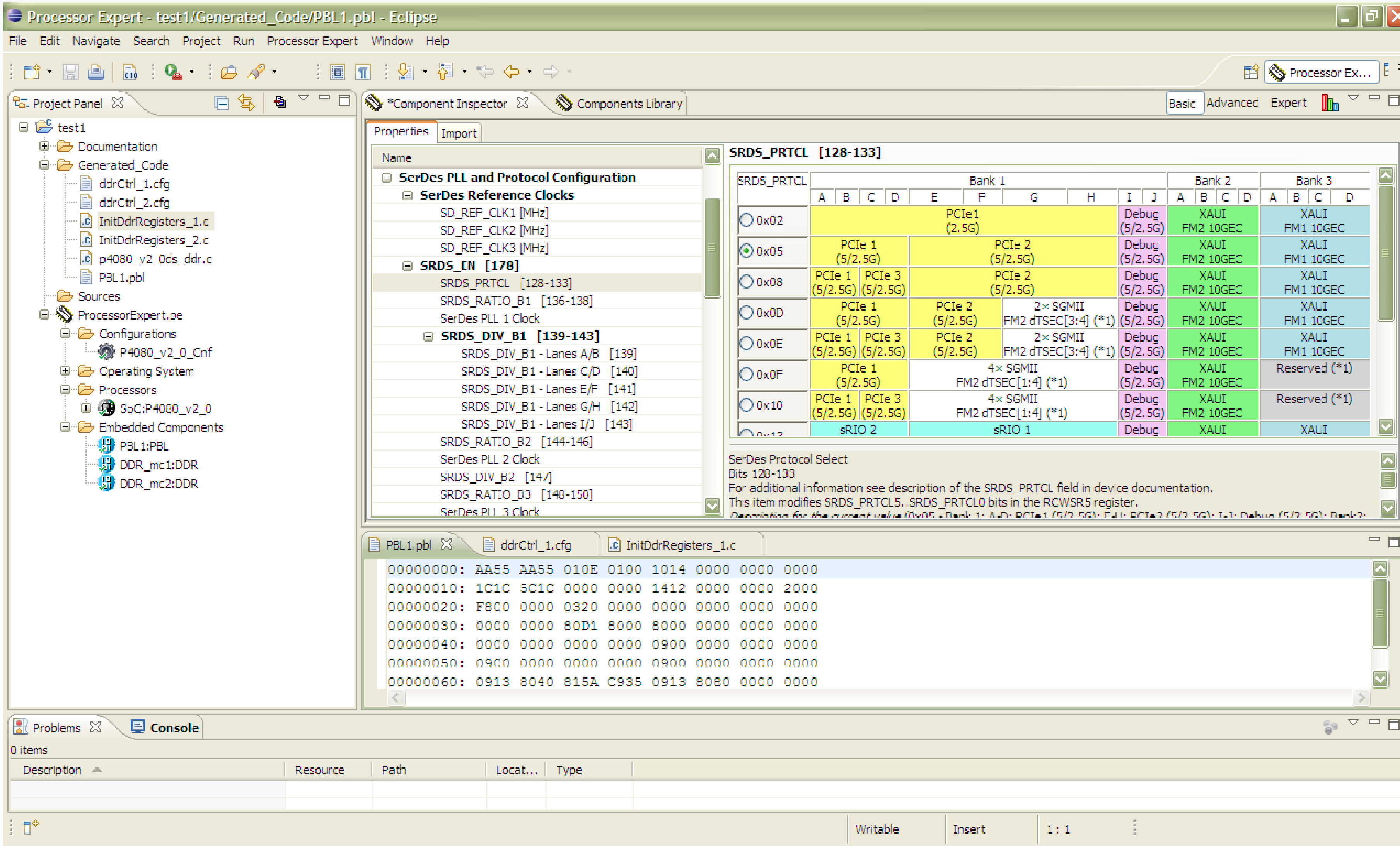
- **QorIQ Configuration Suite v2.3.x is NOW AVAILABLE!!!**
  - Supports all QorIQ and Qorivva devices
  - Works with Eclipse 3.6, Eclipse 3.7 development tools
    - Pure Java solution for maximum choice of host system support
    - Add-in to CodeWarrior Development Studio for PA, v10.1 or later
  - Available from www.freescale.com/QCS – FREE DOWNLOAD*
- **Includes the following configuration tools all designed to collaborate on consistent configuration:**
  - PBL tool to define the Reset Control Word bit values and PBI data for the pre-boot
  - BOOTROM generator for those QorIQ without RCW functionality
  - DDR configuration supports setting the controller to a working state for any DDR
  - Data path graphical view helps to define data path configuration for the DPAA.
  - Hardware Device Tree editor supports references, synchronous GUI and XML editing, node validation based on specification bindings
  - Packaged as a separate product with installer and wizard functionality

* Must be a QorIQ customer or under QorIQ NDA for download permission

Actual URL is http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=PE_QORIQ_SUITE&tid=PEH

freescale™

Processor Expert for QorIQ – Configuration Suite



What's Different About QorIQ Processor Expert?

- **MCUs use Processor Expert to generate source code that is code-size optimized and only includes the minimal functions and operations to support initialization and peripheral drivers**
  - Previously, Processor Expert was included in CodeWarrior only
  - Now, Processor Expert plug-ins can be installed into any Eclipse

- **Processors uses Processor Expert to generate configuration files used in the creation of a bootstrap typically to either Linux or another OS.**
  - Installs as an Eclipse update package (under 20MB)
  - Supports configuration complexity without altering OS / Application software
  - Extensible with on-chip validation tools

freescale™

Installing Processor Expert for QorIQ

- **You need either CodeWarrior for PA 10.1 or later**

  OR, you download an Eclipse version for free

  OR, you use an existing Eclipse workbench you have installed (Wind River, QNX, GNU, etc.)

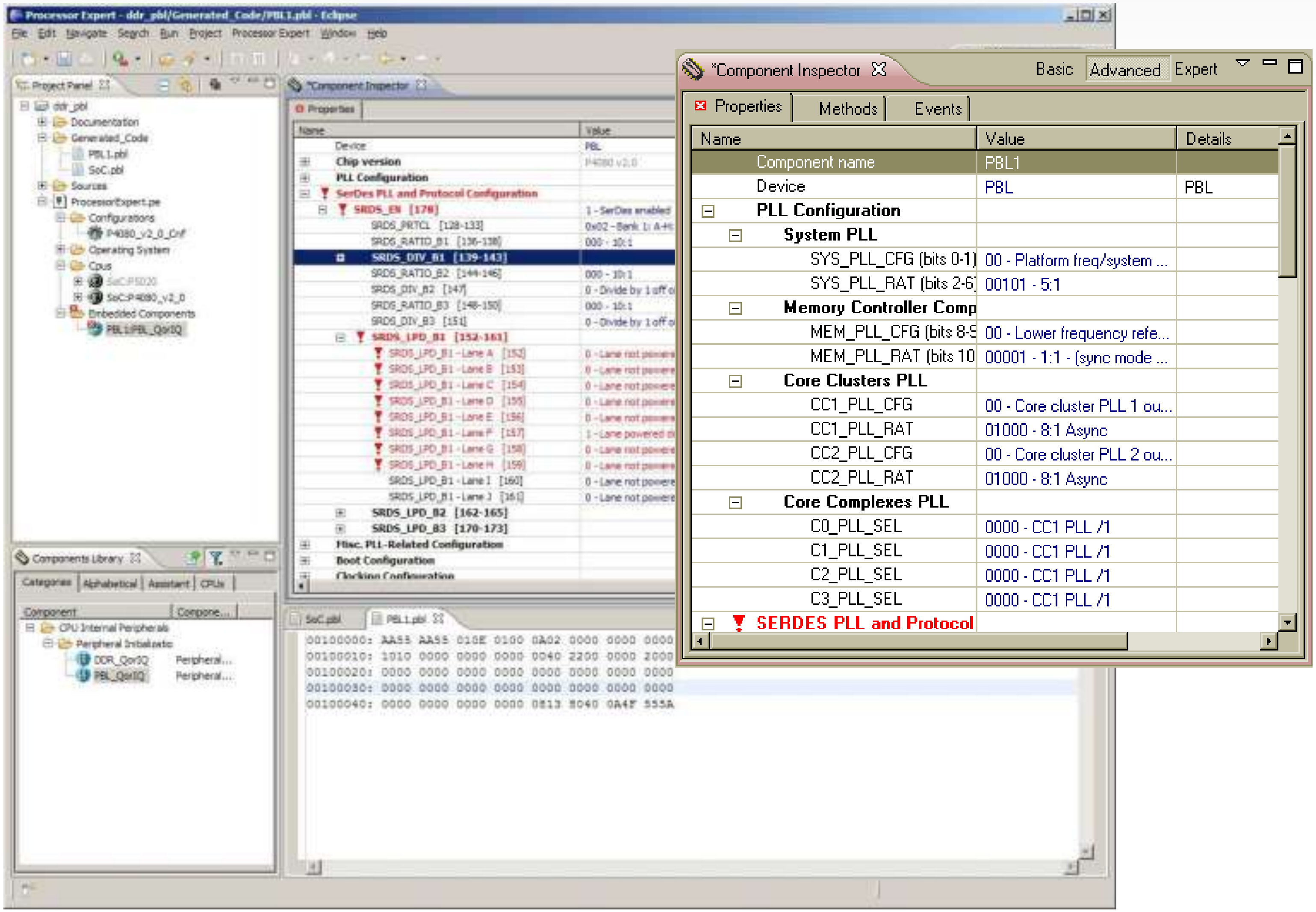
- **Processor Expert for QorIQ Configuration Suite installs using the Eclipse updater's "Add new software…" capability**

- **The Configuration Suite is 100% pure Java so it should run on any Eclipse 3.6.1 or later host environment (Windows, Linux, Solaris, Mac OS, 32-bit/64-bit, …)**

Pre-boot Loader

RCW Configuration Tool

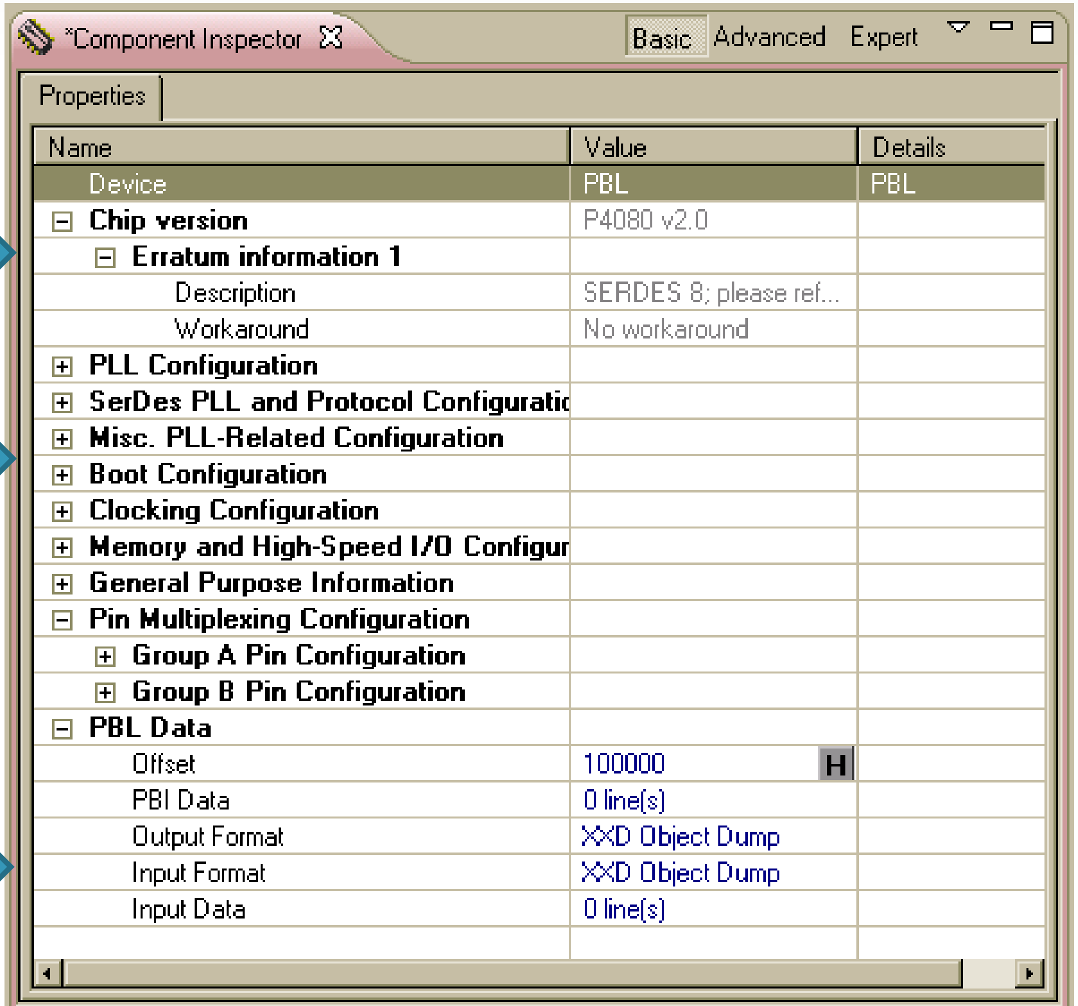
Pre-boot Loader (RCW) Configuration



freescale™

Pre-boot Loader standard component interface

- Chip version and errata information
- Settings of RCW fields
- Input/Output format selection
- Possibility to add PBI data
- Possibility to import RCW settings

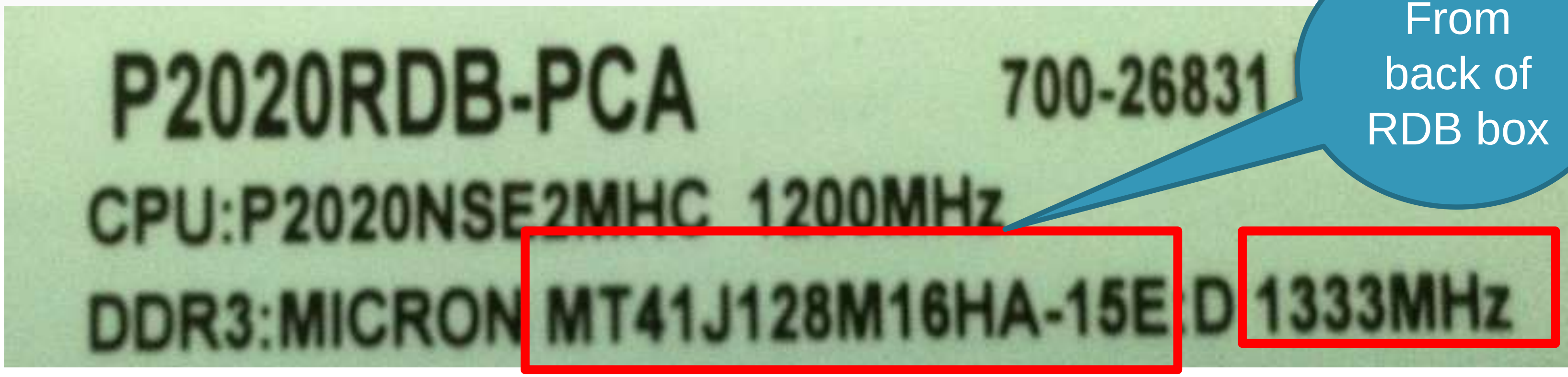


freescale™

DDR Configuration

DDR Configuration Tool

Get DRAM information – P2020RDB-PCA



From back of RDB box

Micron

DDR3 SDRAM

MT41J512M4 – 64 Meg x 4 x 8 Banks

MT41J256M8 – 32 Meg x 8 x 8 Banks

MT41J128M16 – 16 Meg x 16 x 8 Banks

From DRAM datasheet

How about rest of the timing parameters?

Table 1: Key Timing Parameters

Speed Grade	Data Rate (MT/s)	Target ^tRCD-^tRP-CL	^tRCD (ns)	^tRP (ns)	CL (ns)
-093[1, 2, 3, 4]	2133	14-14-14	13.09	13.09	13.09
-107[1, 2, 3]	1866	13-13-13	13.91	13.91	13.91
-125[1, 2,]	1600	11-11-11	13.75	13.75	13.75
-15E[1,]	1333	9-9-9	13.5	13.5	13.5
-187E	1066	7-7-7	13.1	13.1	13.1

Features

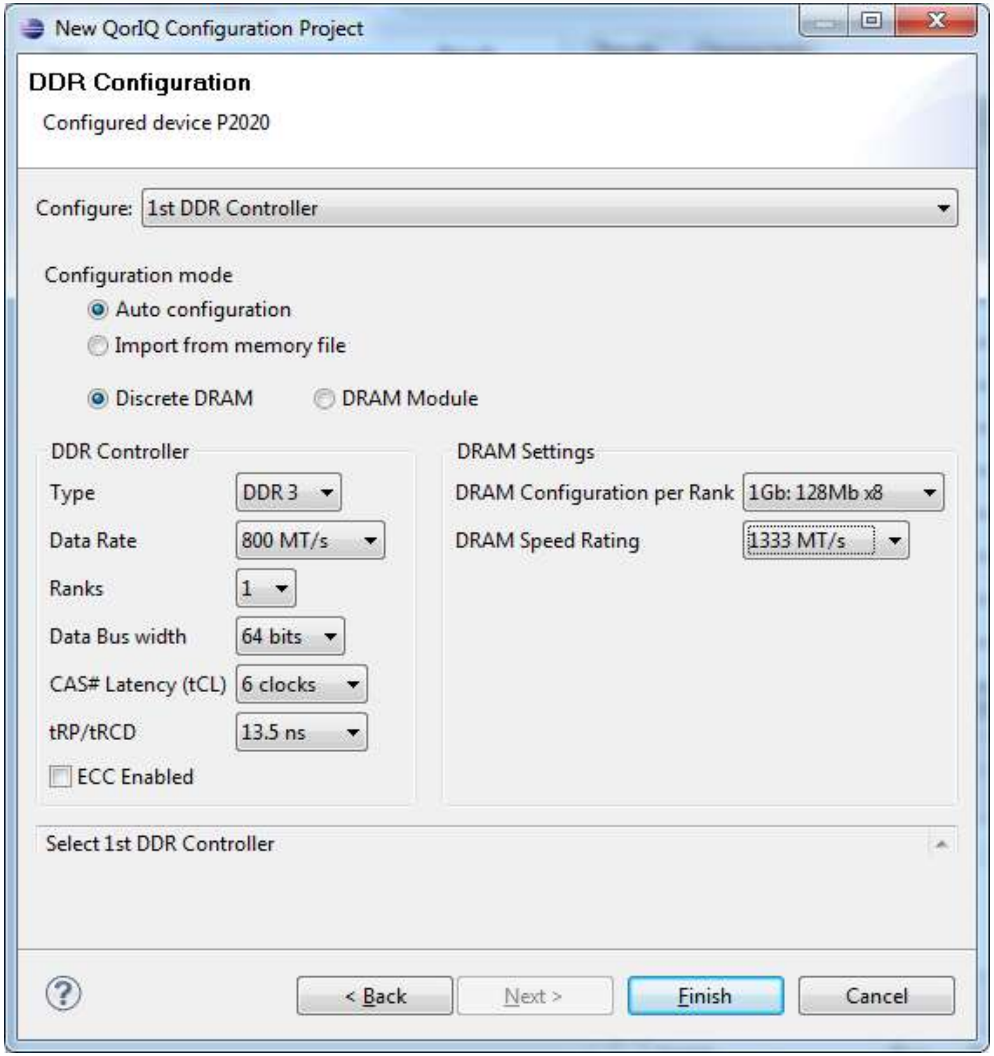
- $V_{DD} = V_{DDQ}$ = 1.5V ±0.075V
- 1.5V center-terminated push/pull I/O
- Differential bidirectional data strobe
- 8*n*-bit prefetch architecture
- Differential clock inputs (CK, CK#)
- 8 internal banks
- Nominal and dynamic on-die termination (ODT) for data, strobe, and mask signals
- Programmable CAS READ latency (CL)
- Posted CAS additive latency (AL)
- Programmable CAS WRITE latency (CWL) based on ^tCK
- Fixed burst length (BL) of 8 and burst chop (BC) of 4 (via the mode register set [MRS])
- Selectable BC4 or BL8 on-the-fly (OTF)
- Self refresh mode
- T_C of 0°C to 95°C
  - 64ms, 8192 cycle refresh at 0°C to 85°C
  - 32ms, 8192 cycle refresh at 85°C to 95°C
- Self refresh temperature (SRT)
- Write leveling
- Multipurpose register
- Output driver calibration

Options[1]	Marking
• Configuration	
– 512 Meg x 4	512M4
– 256 Meg x 8	256M8
– 128 Meg x 16	128M16
• FBGA package (Pb-free) – x4, x8	
– 78-ball (8mm x 10.5mm) Rev. H,M,J,K	DA
– 78-ball (9mm x 11.5mm) Rev. D	HX
• FBGA package (Pb-free) – x16	
– 96-ball (9mm x 14mm) Rev. D	HA
– 96-ball (8mm x 14mm) Rev. K	JT
• Timing – cycle time	
– 938ps @ CL = 14 (DDR3-2133)	-093
– 1.071ns @ CL = 13 (DDR3-1866)	-107
– 1.25ns @ CL = 11 (DDR3-1600)	-125
– 1.5ns @ CL = 9 (DDR3-1333)	-15E
– 1.87ns @ CL = 7 (DDR3-1066)	-187E
• Operating temperature	
– Commercial (0°C ≤ T_C ≤ +95°C)	None
– Industrial (–40°C ≤ T_C ≤ +95°C)	IT
• Revision	:D/:H/:J/:K/:M

Note: 1. Not all options listed can be combined to define an offered product. Use the part catalog search on http://www.micron.com for available offerings.

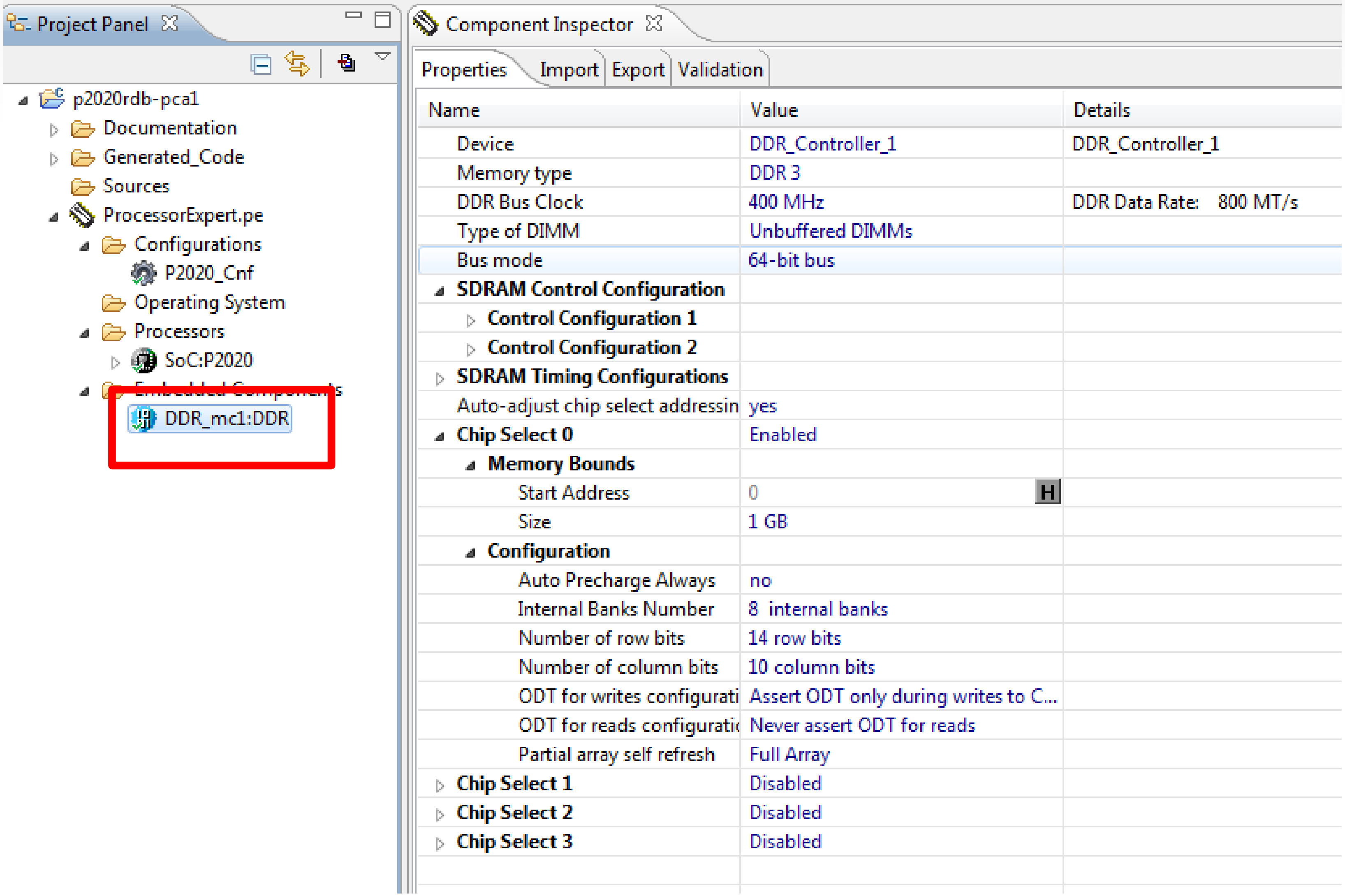
- Tool automatically computes tRCD, tRP, and CL!
  - User can change these values if required.

DDR Wizard simplifies configuration



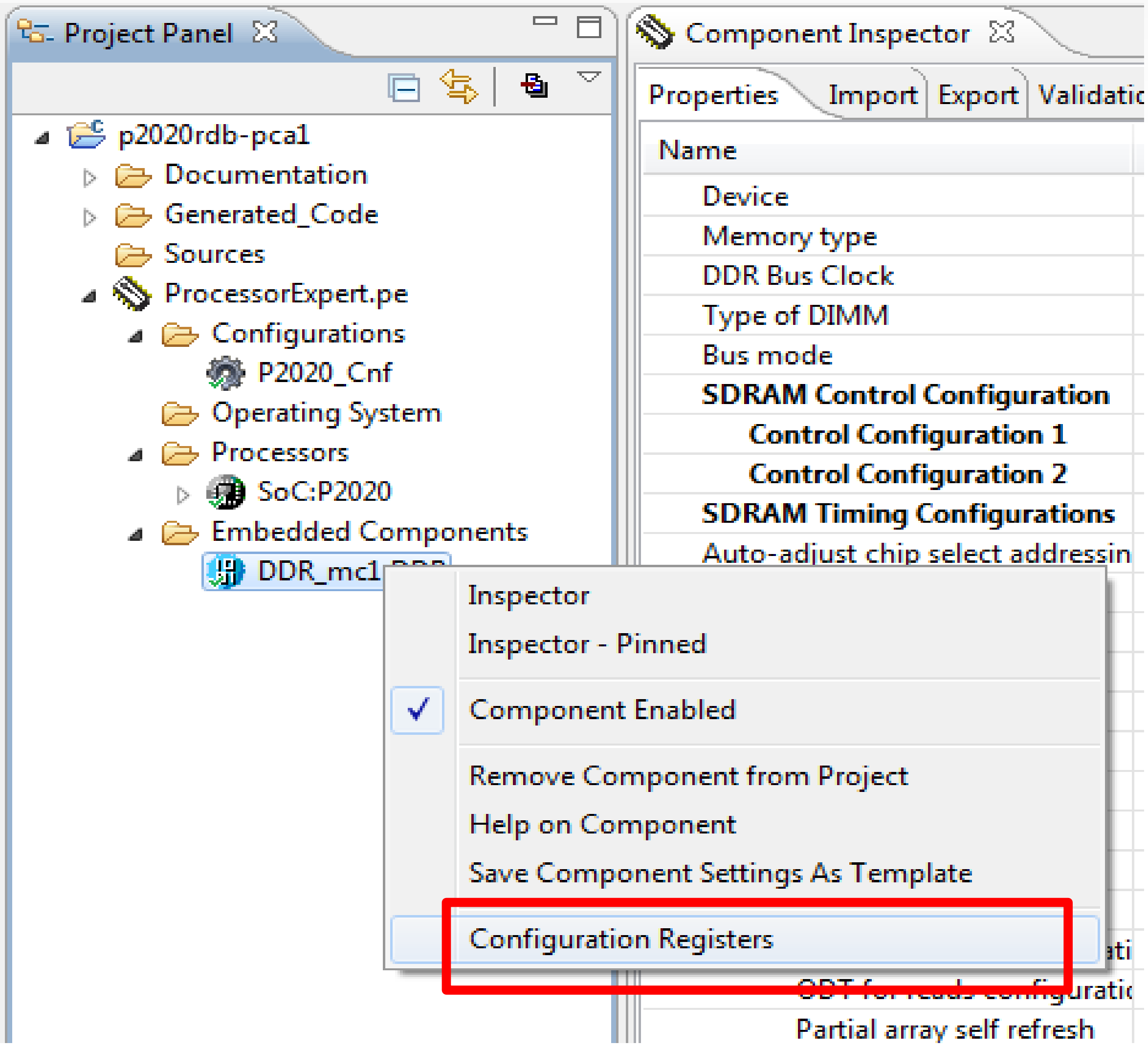
- From memory data sheet:
  - Maximum speed rating
  - Capacity

QCS project explorer



freescale™

Review DDR registers values



Review DDR registers values – contd.

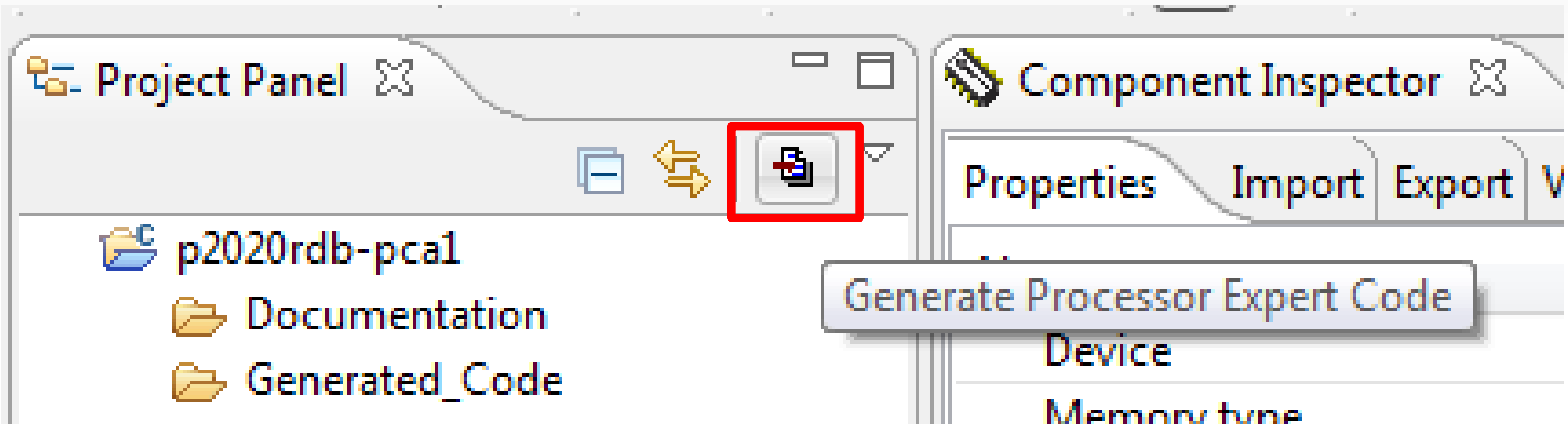
Configuration Registers

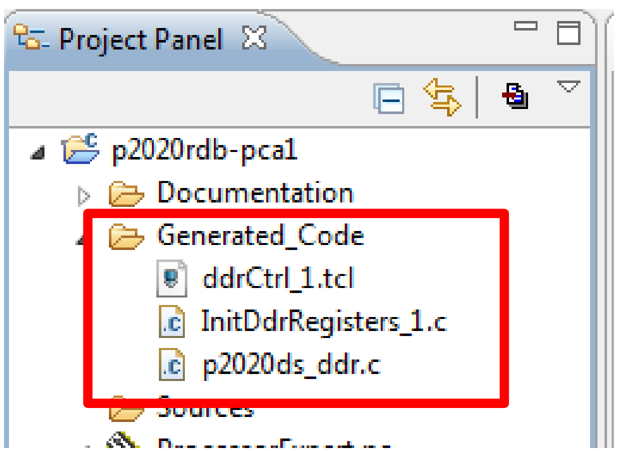
DDR_Controller_1

Reg. name	Init. value	After reset
Peripheral registers		
DDR1_CS0_BNDS	0000003F	00000000
DDR1_CS1_BNDS	00000000	00000000
DDR1_CS2_BNDS	00000000	00000000
DDR1_CS3_BNDS	00000000	00000000
DDR1_CS0_CONFIG	80014202	00000000
DDR1_CS1_CONFIG	00000000	00000000
DDR1_CS2_CONFIG	00000000	00000000
DDR1_CS3_CONFIG	00000000	00000000
DDR1_CS0_CONFIG_2	00000000	00000000
DDR1_CS1_CONFIG_2	00000000	00000000
DDR1_CS2_CONFIG_2	00000000	00000000
DDR1_CS3_CONFIG_2	00000000	00000000
DDR1_TIMING_CFG_3	00030000	00000000
DDR1_TIMING_CFG_0	00330104	00110105
DDR1_TIMING_CFG_1	6E6B8846	00000000
DDR1_TIMING_CFG_2	0FA8D0CC	00000000
DDR1_SDRAM_CFG	47000008	03000000
DDR1_SDRAM_CFG_2	24401050	00000000
DDR1_SDRAM_MODE	00061421	00000000

freescale™

Generate DDR configuration





Generated files – CW, uboot, ddrinit.c

```
# DDR Controller 1 Registers

  # DDR_SDRAM_CFG
  mem [0xFF702110] = 0x47000008

  # CS0_BNDS
  mem [0xFF702000] = 0x3F

  # CS0_CONFIG
  mem [0xFF702080] = 0x80014202

  # CS0_CONFIG_2
  mem [0xFF7020C0] = 0x00

  # TIMING_CFG_3
  mem [0xFF702100] = 0x00030000

  # TIMING_CFG_0
  mem [0xFF702104] = 0x00330104

  # TIMING_CFG_1
  mem [0xFF702108] = 0x6E6B8846

  # TIMING_CFG_2
  mem [0xFF70210C] = 0x0FA8D0CC

  # DDR_SDRAM_CFG_2
  mem [0xFF702114] = 0x24401050

  # DDR_SDRAM_MODE
  mem [0xFF702118] = 0x00061421
```

```
#define DDR_1_INIT_EXT_ADDR_ADDR     0xFF70214C
#define DDR_1_SDRAM_RCW_1_ADDR       0xFF702180
#define DDR_1_SDRAM_RCW_2_ADDR       0xFF702184
#define DDR_1_DATA_INIT_ADDR         0xFF702128
#define DDR_1_SDRAM_MD_CNTL_ADDR     0xFF702120
#define DDR_1_DDRCDR_1_ADDR          0xFF702B28
#define DDR_1_DDRCDR_2_ADDR          0xFF702B2C

#define SDRAM_CFG_MEM_EN_MASK        0x80000000
#define SDRAM_CFG2_D_INIT_MASK       0x00000010

/* DDR Controller configured registers' values */
#define DDR_1_CS0_BNDS_VAL          0x3F
#define DDR_1_CS1_BNDS_VAL          0x00
#define DDR_1_CS2_BNDS_VAL          0x00
#define DDR_1_CS3_BNDS_VAL          0x00
#define DDR_1_CS0_CONFIG_VAL        0x80014202
#define DDR_1_CS1_CONFIG_VAL        0x00
```

```
#define PEX_CONFIG_DDR1_INIT_EXT_ADDR      0x00000000
#define PEX_CONFIG_DDR1_TIMING_4           0x00220001
#define PEX_CONFIG_DDR1_TIMING_5           0x02401400
#define PEX_CONFIG_DDR1_ZQ_CNTL            0x89080600
#define PEX_CONFIG_DDR1_WRLVL_CNTL         0x8655F614
#define PEX_CONFIG_DDR1_RCW_1              0x00000000
#define PEX_CONFIG_DDR1_RCW_2              0x00000000


/* DDR Controller 1 configuration global structures */
fsl_ddr_cfg_regs_t ddr_cfg_regs_0 = {
  .cs[0].bnds = PEX_CONFIG_DDR1_CS0_BNDS,
  .cs[1].bnds = PEX_CONFIG_DDR1_CS1_BNDS,
  .cs[2].bnds = PEX_CONFIG_DDR1_CS2_BNDS,
  .cs[3].bnds = PEX_CONFIG_DDR1_CS3_BNDS,
  .cs[0].config = PEX_CONFIG_DDR1_CS0_CONFIG,
  .cs[1].config = PEX_CONFIG_DDR1_CS1_CONFIG,
  .cs[2].config = PEX_CONFIG_DDR1_CS2_CONFIG,
```

Device Tree Editor

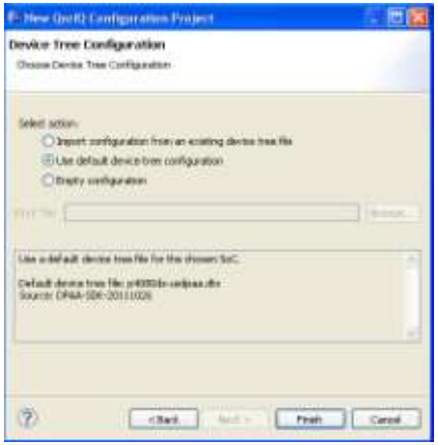
Hardware Device Tree Tool



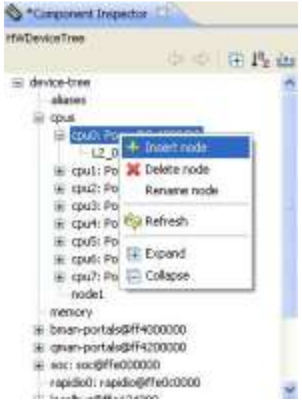


Hardware Device Tree Workflow

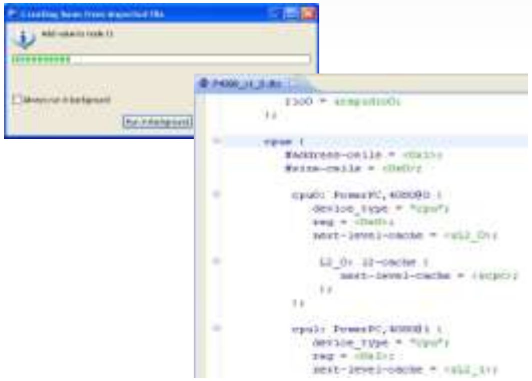
Create Project



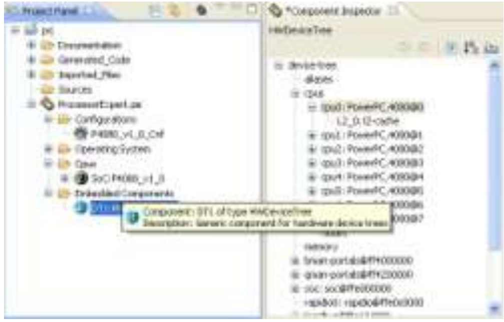
Configure Component



Generate Code



Select Component

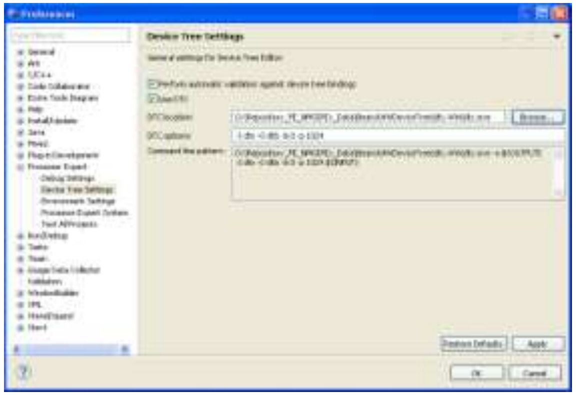


Validate Component



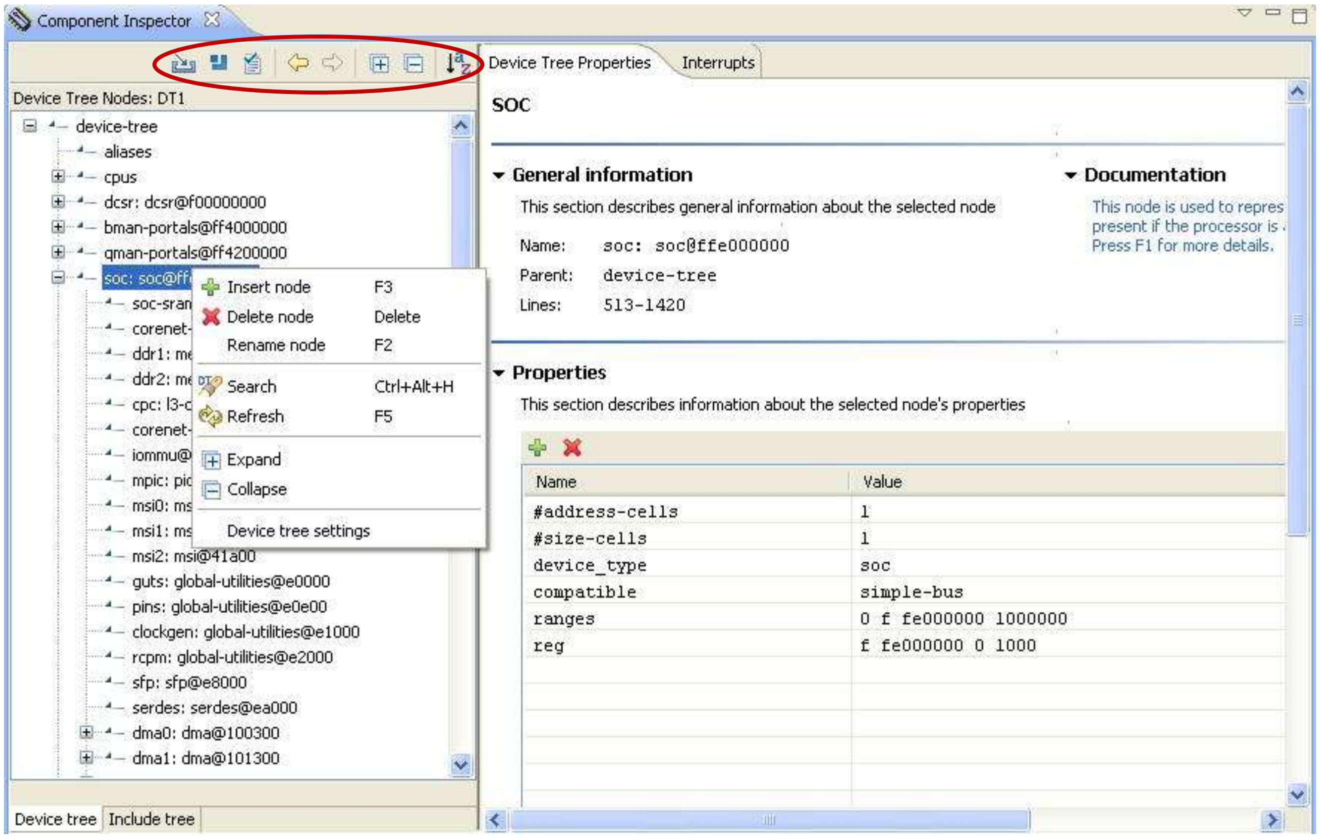
Compile DTS

```
#!/bin/bash
dtc -f -b 0 -p 0x8000 -R 8 -I dts -o dtb $1.dts
```



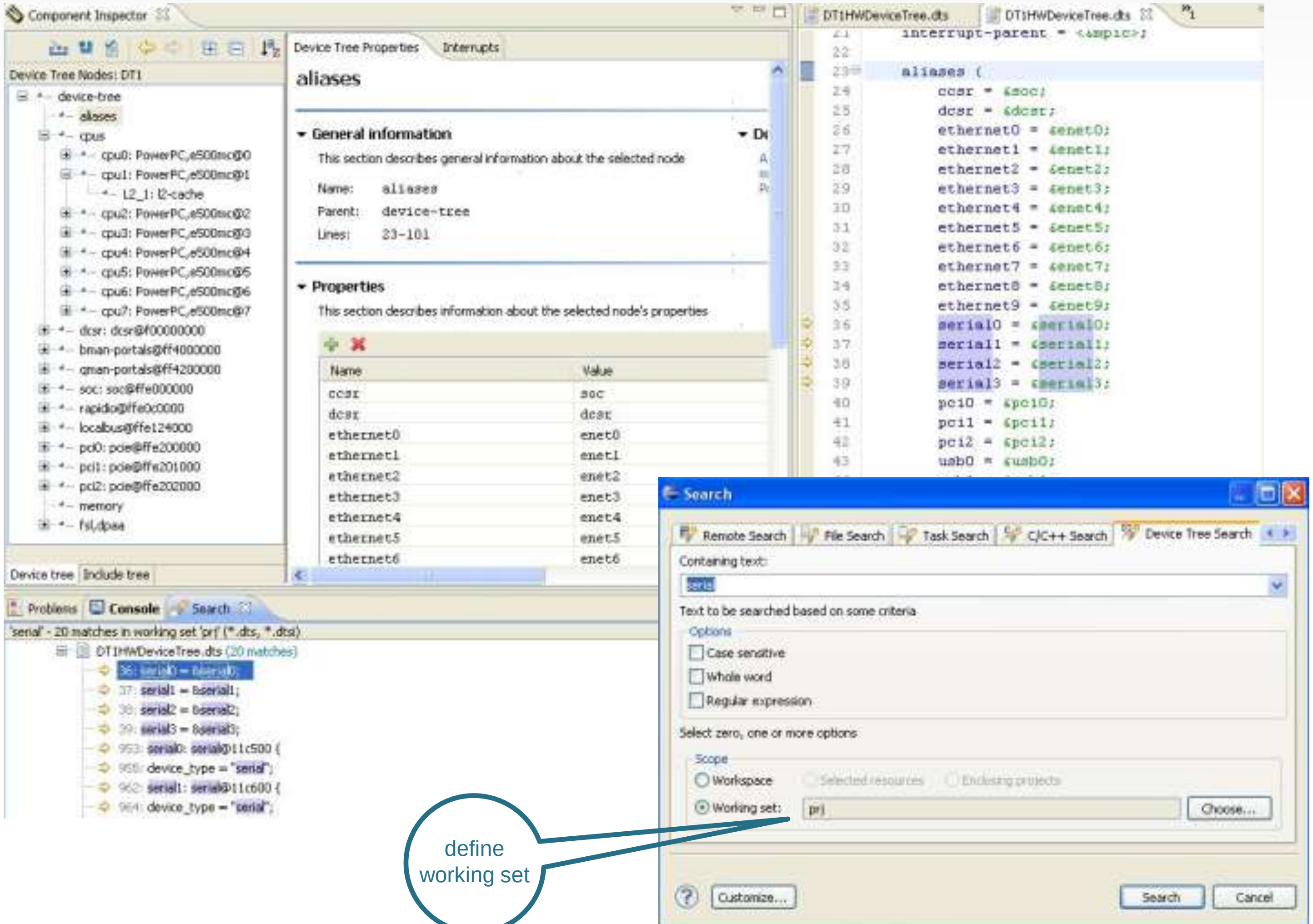
Explorer Tree View

- Operations on nodes
  - Go back / forward
  - Expand/collapse
  - Ascending/descending sort
  - Insert node
  - Delete node
  - Rename node
- Other operations
  - Import device tree
  - Include device tree
  - Validate device tree
  - Search in device tree



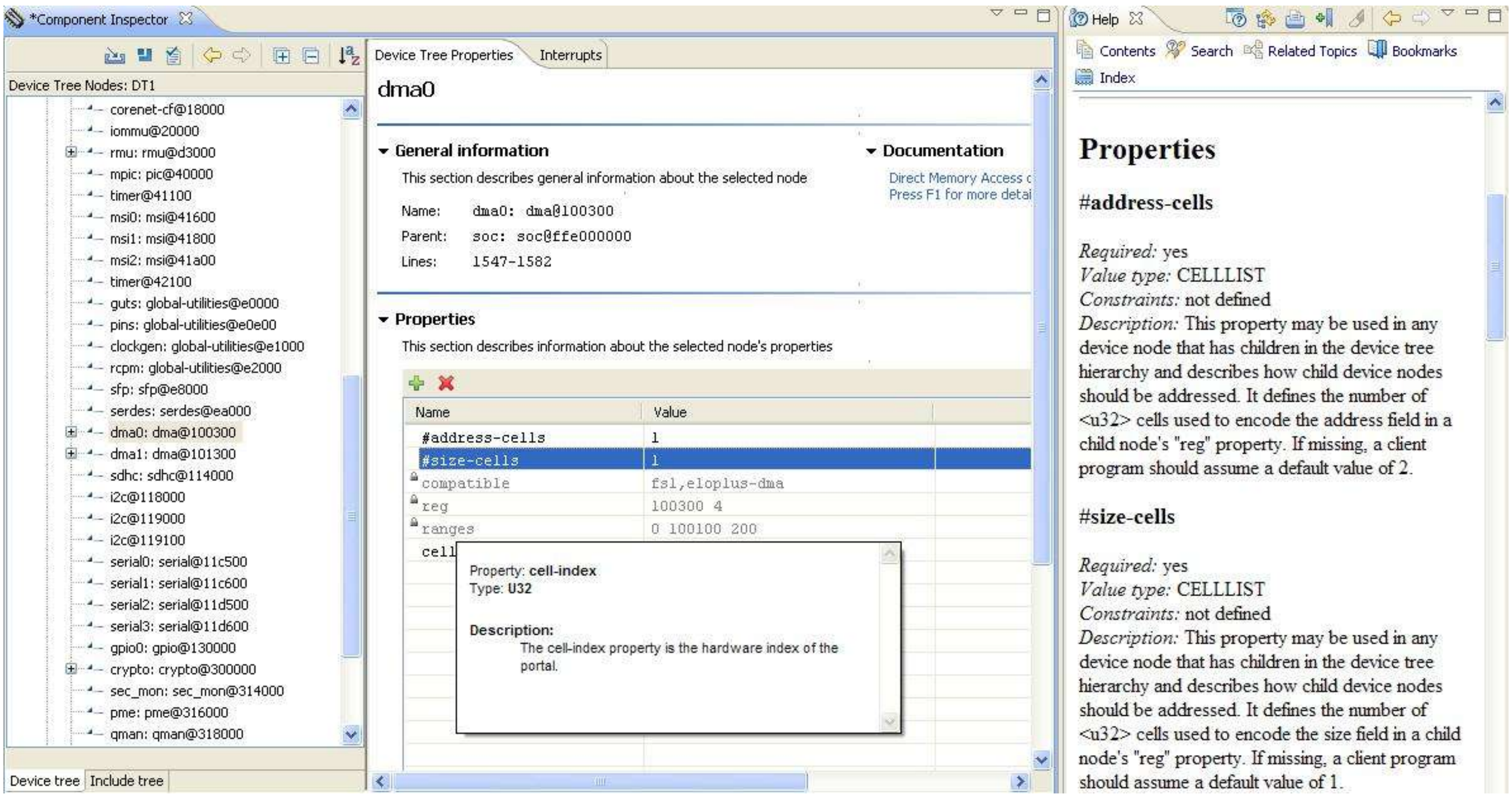
freescale™

Search Capability



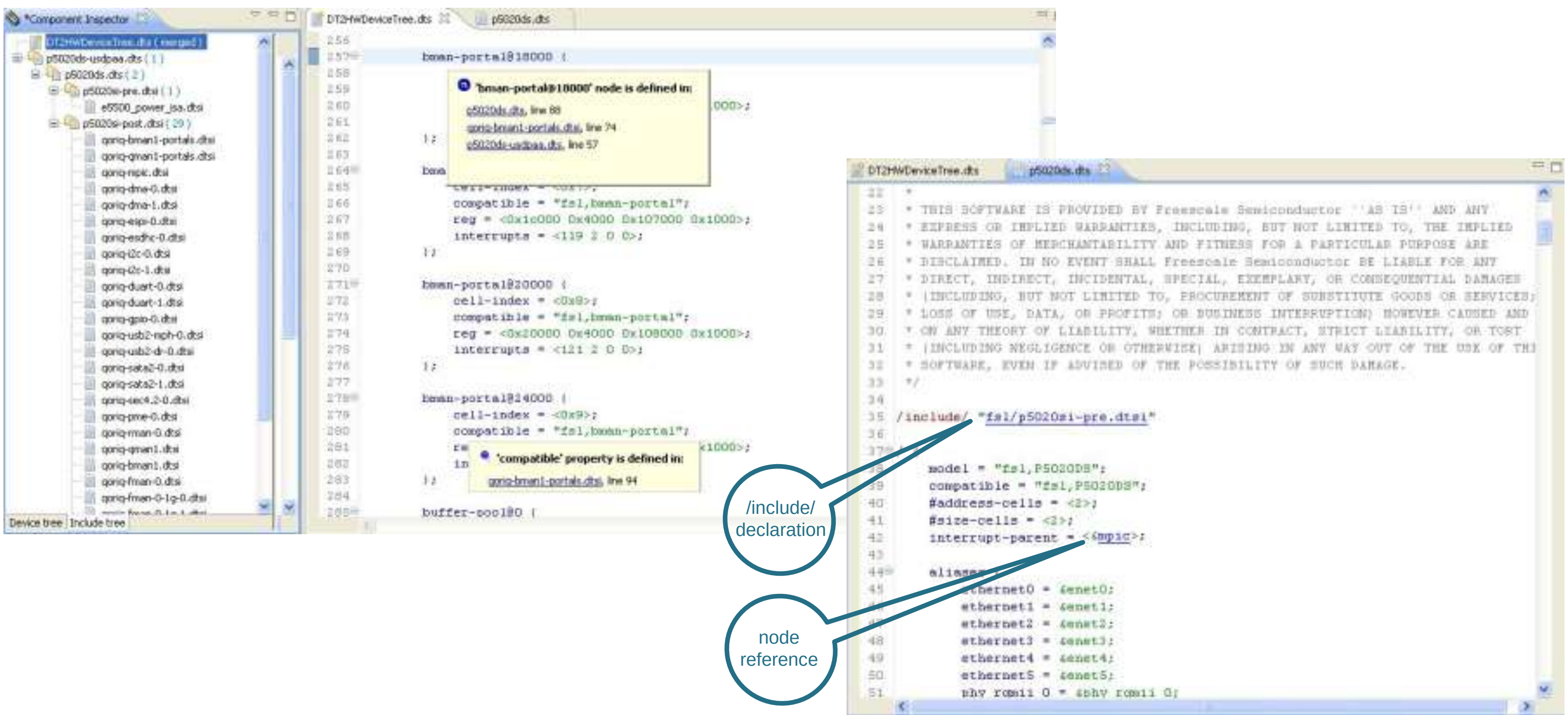
Device Tree Bindings

- Each node has a “binding” representing its schema. It describes what properties are optional or required and what each means.



Device trees inclusion

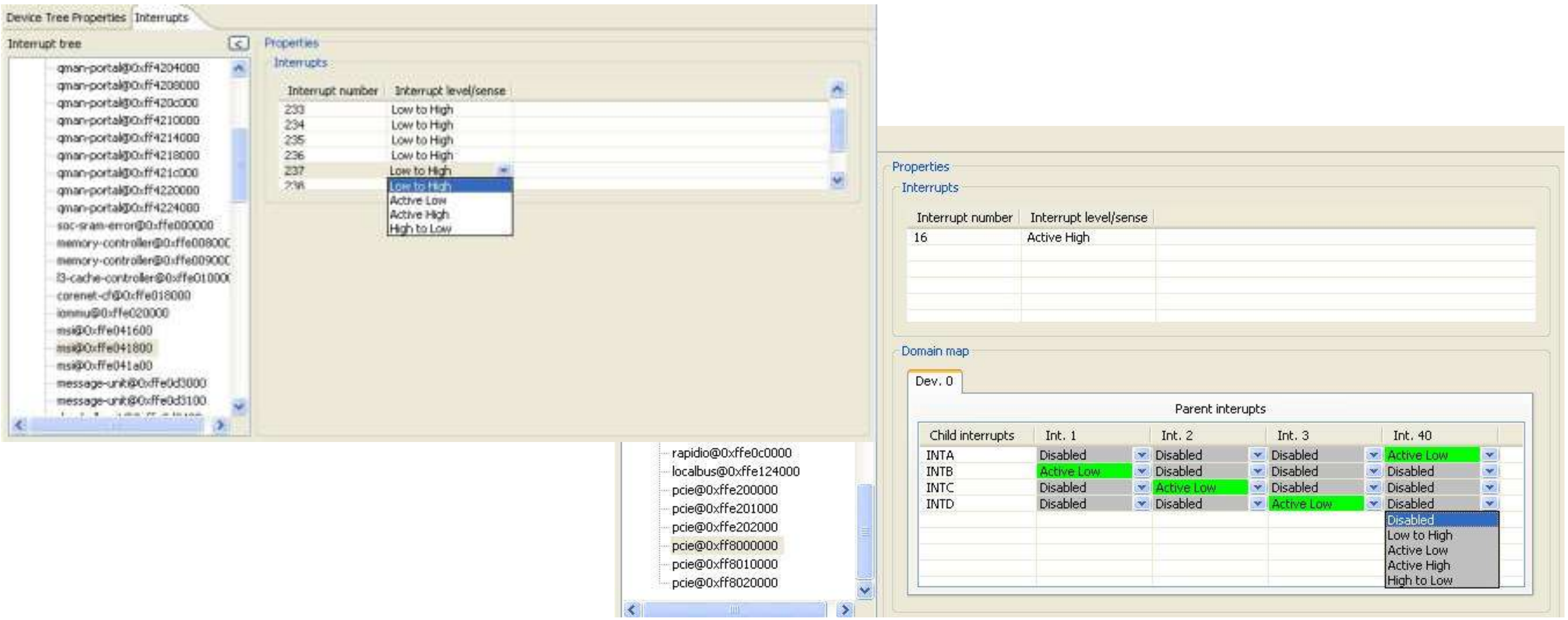
- The Include tree allows easy navigation among device tree fragments (dts, dtsi).
- Hovering support for properties and nodes: a tool-tip appears displaying their initial locations.
- Hyperlink detection for /include/ declarations and device tree references (Ctrl + left click).



freescale™

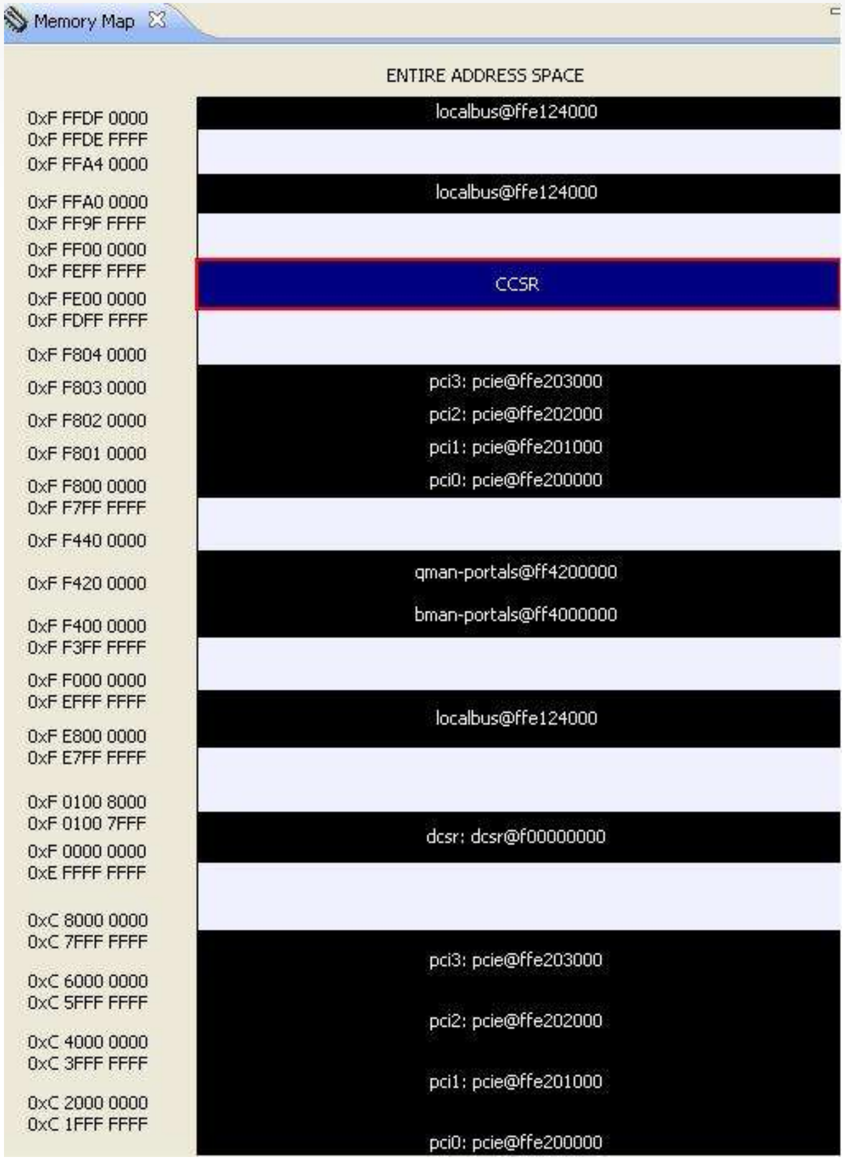
Interrupts tree

- The Interrupts tree represents the hierarchy and routing of interrupts in the platform hardware.
- The left side displays the actual representation of the Interrupt tree starting from the root interrupt controller.
- The right side displays the interrupts sources for the selected device tree node.



Memory Map view

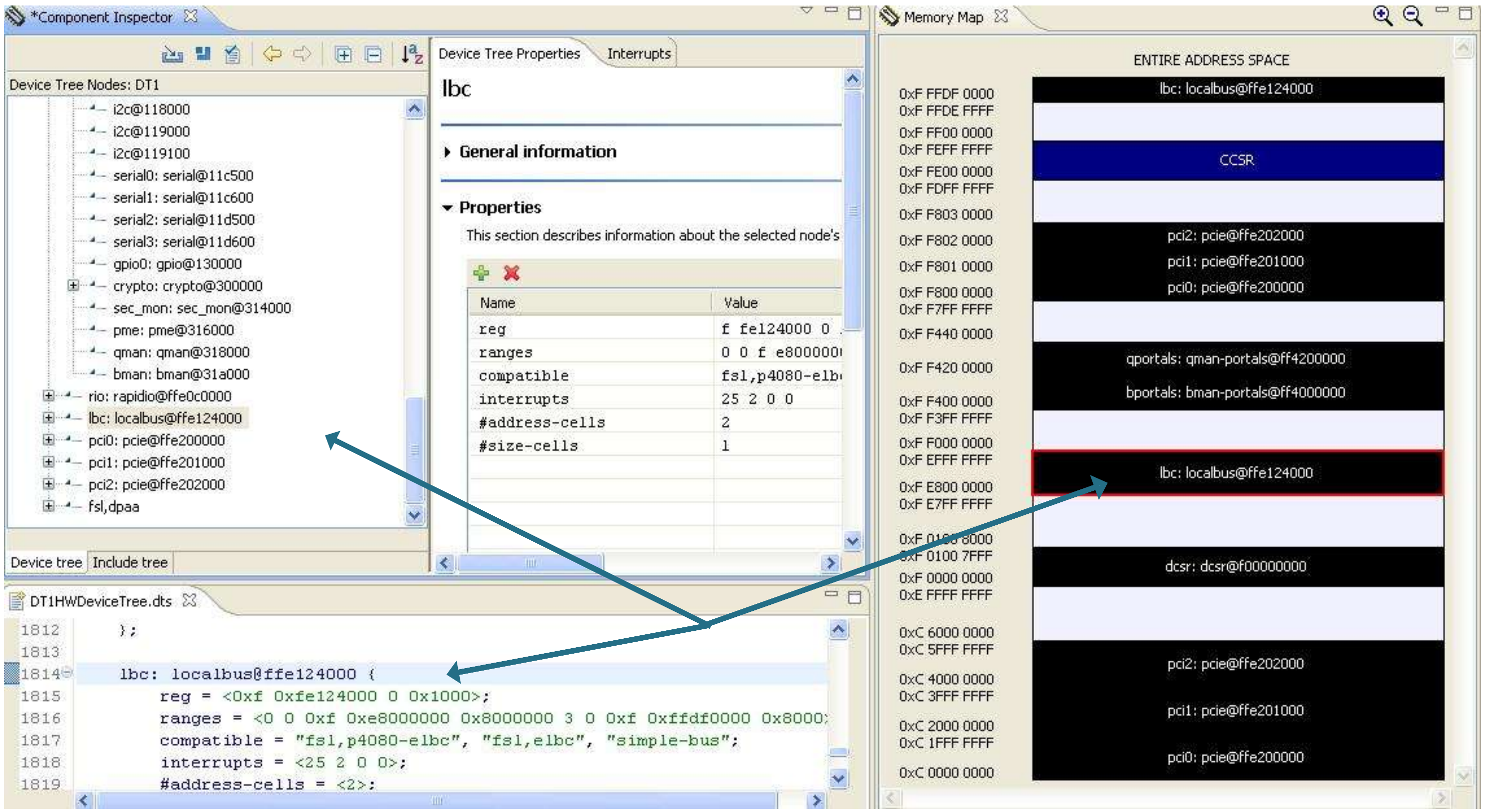
- Any hw device tree can be seen as a representation of different Local Access Windows (LAW).
- Each LAW maps to a specified target interface, such as DDR Controller, Localbus, PCI Express, etc.
- Each device tree node having reg and ranges properties defines a memory range inside/outside Configuration Control and Status Register (CCSR) space area.
- The Memory Map view pops-up automatically when a device tree component is selected inside Component Inspector view.



freescale™

Device tree views synchronization

- Device tree views
- GUI <=> text editor symmetry
- Memory map view => GUI editor symmetry
- Modifications are reflected in all editors



freescale™

DPAA Configuration

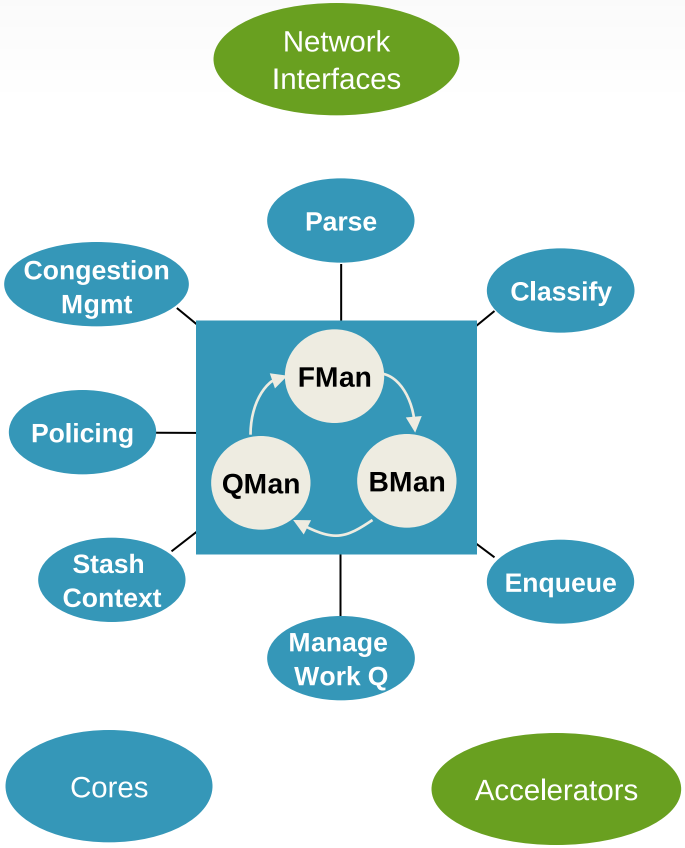
Data Path Graphing Tool





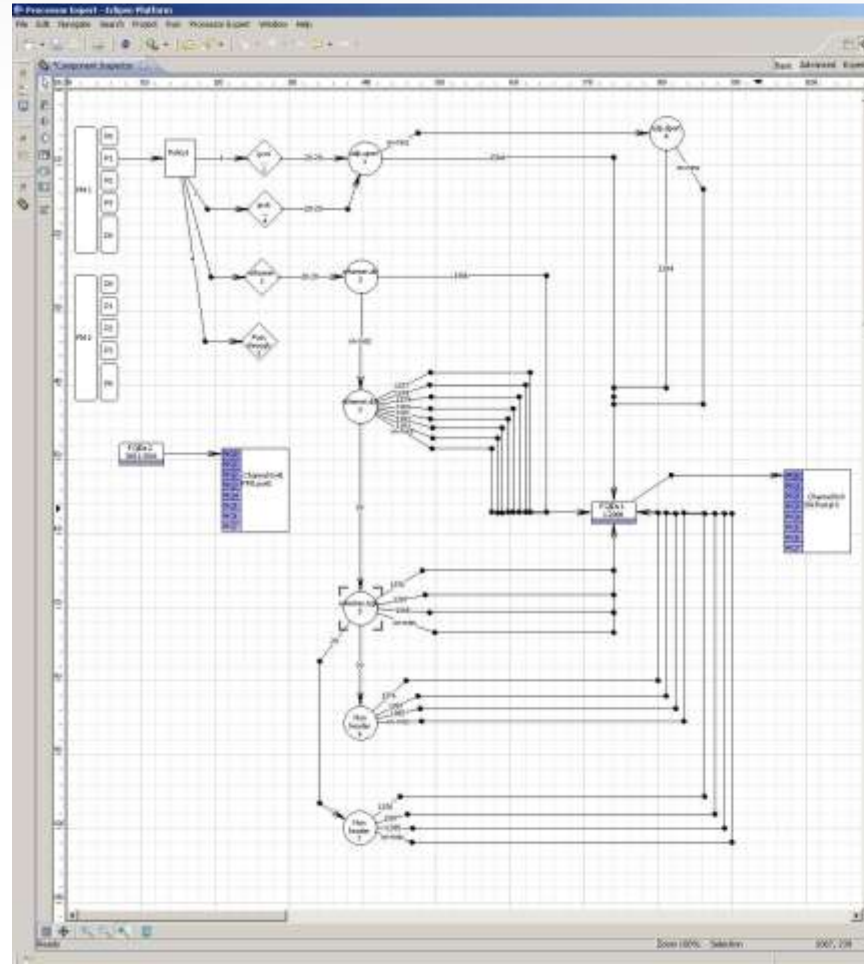
DPAA Overview

- **DPAA (Data Path Acceleration Architecture), provides the infrastructure to pass packets to/from cores, hardware accelerators and network interfaces**
- **The architecture contains several hardware components**
  - Frame Manager (FM)
  - Buffer Manager (BM)
  - Queue Manager (QM)
  - HW accelerators: Security (SEC), Pattern Matching Engine (PME)
- **Each hardware component is performing specific operations on the incoming/outgoing frames**
  - BM – Manages data storage buffer pools. Is a shared resource among cores, network interfaces, and HW accelerators
  - FM – supports in-line/off-line packet parsing and initial classification. It enables policing and flow and QoS – based packet distribution to the cores
  - QM – Manages the queuing of data between cores, network interfaces, and HW accelerators
  - SEC – provides cryptographic acceleration
  - PME – high performance hardware pattern matching functionality



QCS DPAA Component Overview

- Flowchart representation of DPAA component is a software solution intended to ease creation of complex DPAA configurations
- Have an intuitive graphical representation
- Easy to understand the overall architecture as well as individual DPAA components
- QCS integrated component designed to ease DPAA configuration for QorIQ devices
- Interactive and user friendly interface in order to provide the best user experience
- Allows customers to easily translate their own data flow into a valid driver configuration
- Designed to deal with complex DPAA user scenarios



```xml
<?xml version="1.0" encoding="utf-8"?>

<dpc xmlns:xi="http://www.w3.org/2001/XInclude">

<cfgdata>
	<config>
		<engine name="fm0">
			<port type="1G" number="0" policy="L2TPv3" portid="0x0"/>
			<port type="1G" number="3" policy="UDP" portid="0x0"/>
		</engine>
		<engine name="fm1"/>
	</config>
</cfgdata>

<netpcd>
	<distribution name="Pass-through">
		<queue count="16" base="0x85"/>
	</distribution>
	<distribution name="IPv4">
		<queue count="128" base="0x123"/>
		<protocols>
			<protocolref name="ipv4"/>
			<protocolref name="ipv6"/>
		</protocols>
	</distribution>
	<distribution name="MAC.dst">
		<queue count="128" base="0x980"/>
		<key>
			<fieldref name="ethernet.dst"/>
			<fieldref name="ethernet.src"/>
			<fieldref name="vlan.tci"/>
		</key>
		<action type="classification" name="MAC.dst"/>
	</distribution>
	<distribution name="UDP">
		<queue count="16" base="0x1450"/>
		<protocols>
			<protocolref name="udp"/>
		</protocols>
	</distribution>
	<classification name="UDP">
		<key>
			<fieldref name="ethernet.dst"/>
		</key>
		<entry>
			<data>0x0</data>
			<mask>0xf</mask>
			<queue base="0x1000"/>
		</entry>
		<entry>
			<data>0x0</data>
			<mask>0xf</mask>
			<queue base="0x1100"/>
		</entry>
		<action condition="on-miss" type="drop"/>
	</classification>
	<policy name="L2TPv3">
		<dist_order>
			<distributionref name="IPv4"/>
			<distributionref name="MAC.dst"/>
			<distributionref name="Pass-through"/>
		</dist_order>
	</policy>
	<policy name="UDP">
		<dist_order>
			<distributionref name="UDP"/>
		</dist_order>
	</policy>
</netpcd>

<bman name = "bman_master">
	<portals>
	 <bmportal id = "0" irq = "false"/>
				<bmportal id = "3" irq = "false"/>
	</portals>
	<liodn>20</liodn>
	<irq>false</irq>
</bman>

<bufferpool name = "bpool_1">
	<bpid>1</bpid>
	<buffers>1000</buffers>
			<size>400</size>
</bufferpool>

<bufferpool name = "bpool_2">
	<bpid>2</bpid>
	<buffers>1000</buffers>
			<size>400</size>
</bufferpool>

<qman name = "qman_master">
	<portals>
				<qmportal id = "0" irq = "false" liodn = "1"/>
				<qmportal id = "3" irq = "false" liodn = "1"/>
	</portals>
	<totalfqids>150000</totalfqids>
	<fqdmemorypartition> e_MEM_1ST_DDR_NON_CACHEABLE
</fqdmemorypartition>
	<pfdrmemorypartition> e_MEM_1ST_DDR_NON_CACHEABLE
</pfdrmemorypartition>
	<irq>false</irq>
			<fqBase>1</fqBase>
			<rtframesdepth>30000</rtframesdepth>
			<pfdrtreshold>0</pfdrtreshold>
			<sfdrtreshold>0</sfdrtreshold>
</qman>

<fmport name = "fm0port01" type = "1G" number = "1" engine = "fm0">
	<rx name = "fm0port01rx">
		<errFqid>1</errFqid>
		<dfltFqid>2</dfltFqid>
		<liodnOffset>0</liodnOffset>
		<bufferpools>
			<bpool name = "bpool_1"/>
			<bpool name = "bpool_2"/>
			<bpool name = "bpool_3"/>
		</bufferpools>
	</rx>
	<nonrx name = "fm0port01tx">
		<errFqid>3</errFqid>
		<dfltFqid>4</dfltFqid>
	</nonrx>
	<mac name = "fm0port01mac">
		<irq>false</irq>
		<addr>0x00049f000266</addr>
		<interface>e_ENET_IF_RGMII</interface>
		<speed>e_ENET_SPEED_1000</speed>
		<resetoninit>true</resetoninit>
		<loopback>true</loopback>
	</mac>
	<dpaport name = "dpaportFm0P01"/>
</fmport>

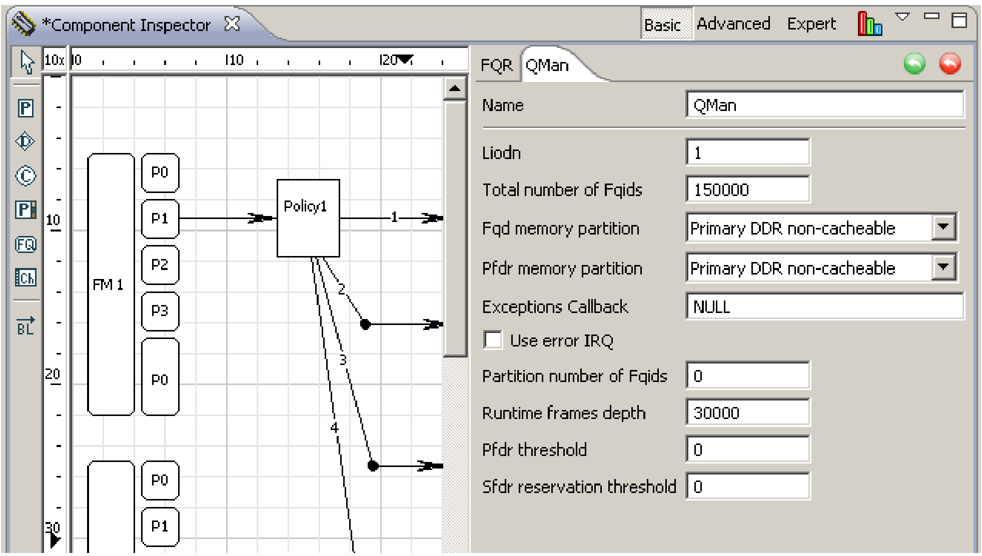
<fmport name = "fm0port03" type = "1G" number = "3" engine = "fm0">
	<rx name = "fm0port03rx">
		<errFqid>1</errFqid>
		<dfltFqid>2</dfltFqid>
		<liodnOffset>0</liodnOffset>
		<bufferpools>
			<bpool name = "bpool_1"/>
			<bpool name = "bpool_2"/>
			<bpool name = "bpool_3"/>
		</bufferpools>
	</rx>
	<nonrx name = "fm0port01tx">
		<errFqid>3</errFqid>
		<dfltFqid>4</dfltFqid>
	</nonrx>
	<mac name = "fm0port01mac">
		<irq>false</irq>
		<addr>0x00049f000266</addr>
		<interface>e_ENET_IF_RGMII</interface>
		<speed>e_ENET_SPEED_1000</speed>
		<resetoninit>true</resetoninit>
		<loopback>true</loopback>
	</mac>
	<dpaport name = "dpaportFm0P01"/>
</fmport>

<fqr name = "FQIDs_1-100">
	<dest>0x60</dest>
	<workqueue>0</workqueue>
	<force>true</force>
	<fqid>1</fqid>
	<count>100</count>
	<align>128</align>
	<txconfclbk>FQR_TxConfCB</txconfclbk>
</fqr>

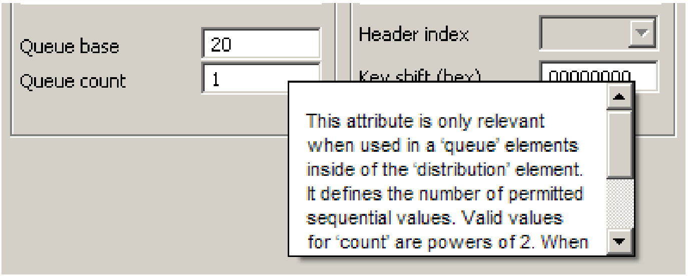
<fqr name = "FQIDs_101-500">
	<dest>0x21</dest>
	<workqueue>2</workqueue>
	<force>true</force>
```

QCS DPAA Component Features (1)

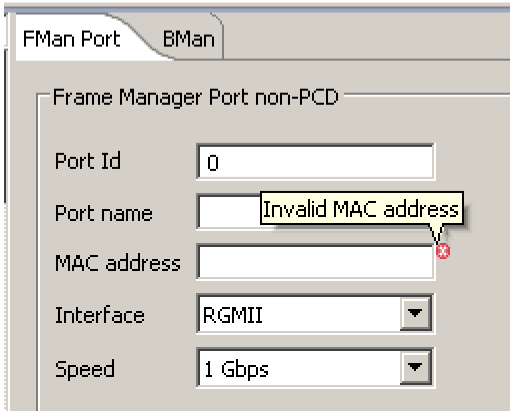
- Default values capability
- Easy access to configuration settings for each DPAA element



Instant display of relevant description for each configuration parameter



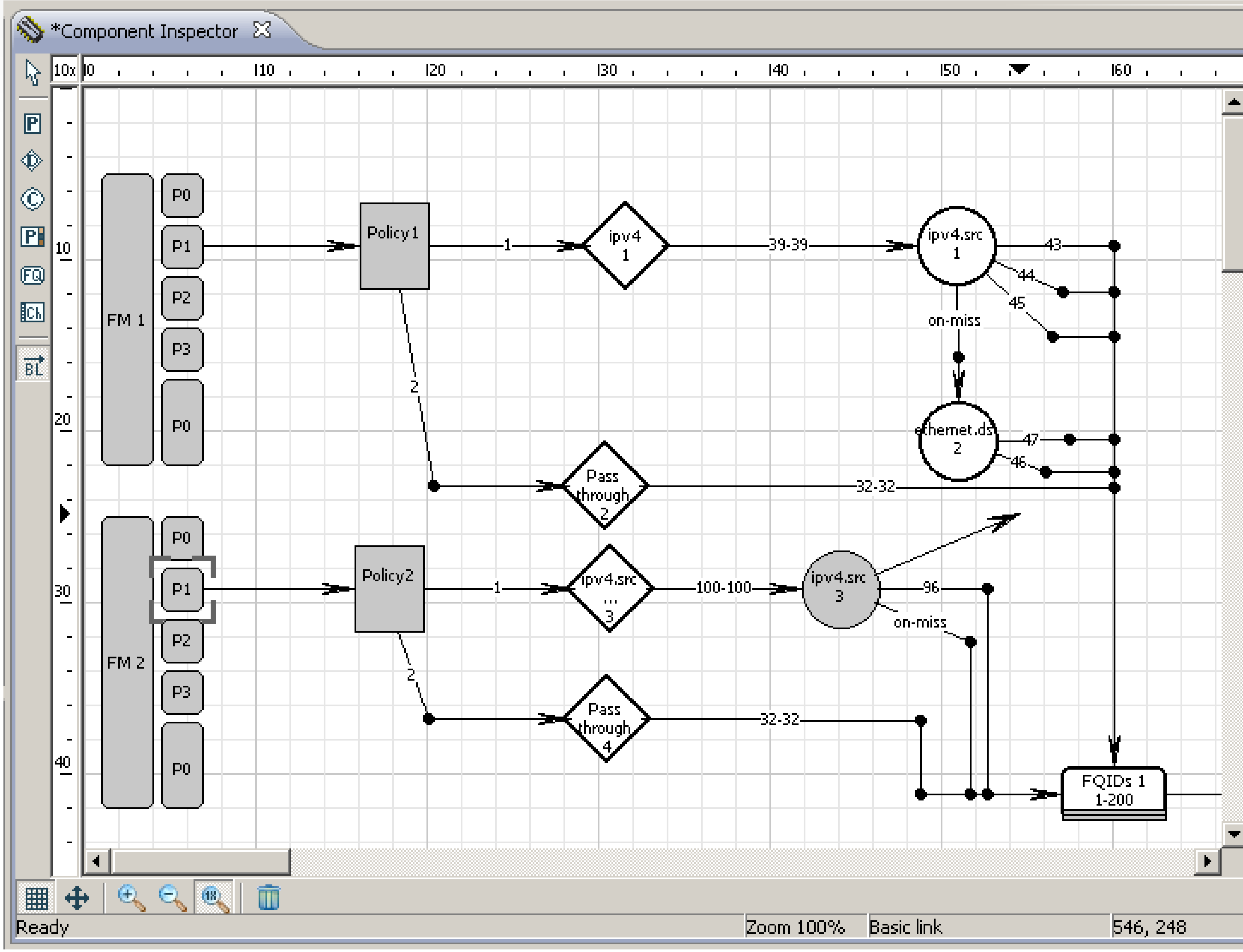
Automatic input validation, configuration constraints checking and instant display of relevant conflict messages



freescale™

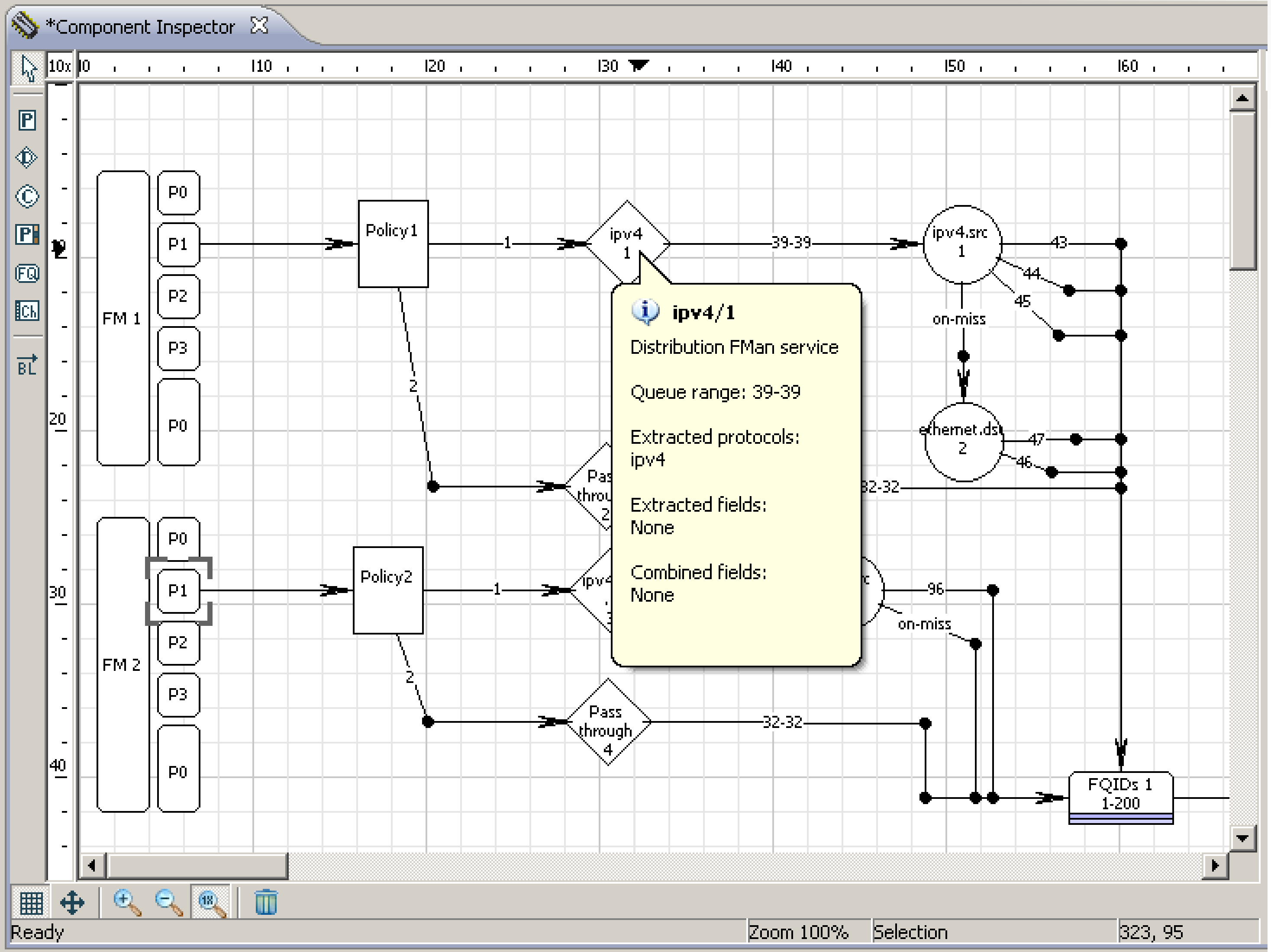
QCS DPAA Component Features (2)

- On-the-fly configuration validation by highlighting correct choices and graying out the invalid ones



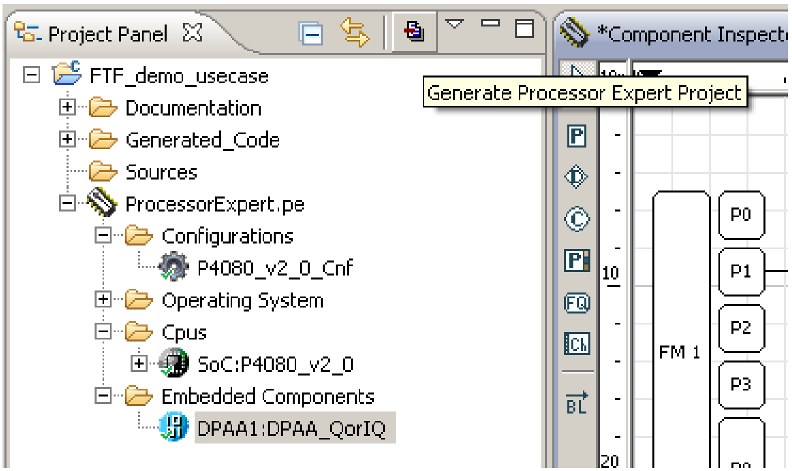
QCS DPAA Component Features (3)

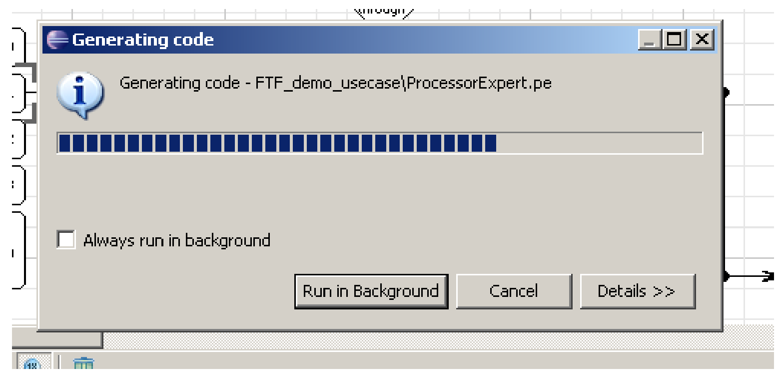
- Instant display of relevant configuration summary for each DPAA element



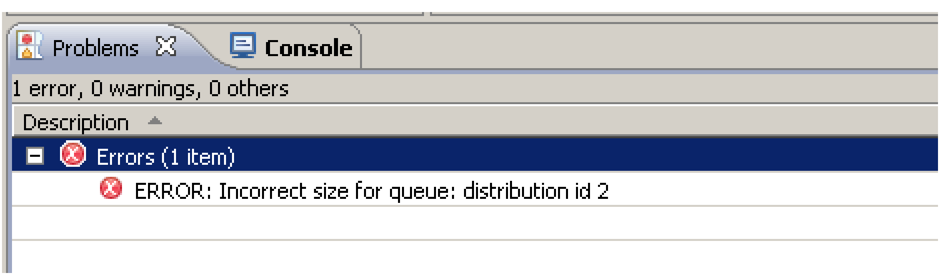
QCS DPAA Component Features (4)

- Immediate code generation at user request in any stage of configuration





- Immediate notification for all errors occurring during the code generation process



Import XMLs Feature

- **Imports DPAA configuration from Freescale extensions to NetPDL xml based files:**
  - DPAA objects
  - Connections between them
  - DPAA objects configuration
  - Automatic update of the objects and links after import is done
- **The xml files can be generated using the QCS solution or can be created by hand**

DPC Configuration

- this is the file that has to be imported

```xml
<?xml version="1.0" encoding="utf-8"?>

<dpc xmlns:xi="http://www.w3.org/2001/XInclude">

<xi:include href="config_guest_0.xml"/>
<xi:include href="pcd_guest_0.xml"/>

<bman name = "bman_master">
            <portals>
                <bmportal id = "0" irq = "false"/>
            </portals>
            <liodn>20</liodn>
            <irq>false</irq>
</bman>

<bufferpool name = "bpool_1">
            <buffers>100</buffers>
            <size>51200</size>
</bufferpool>
```

PCD Configuration

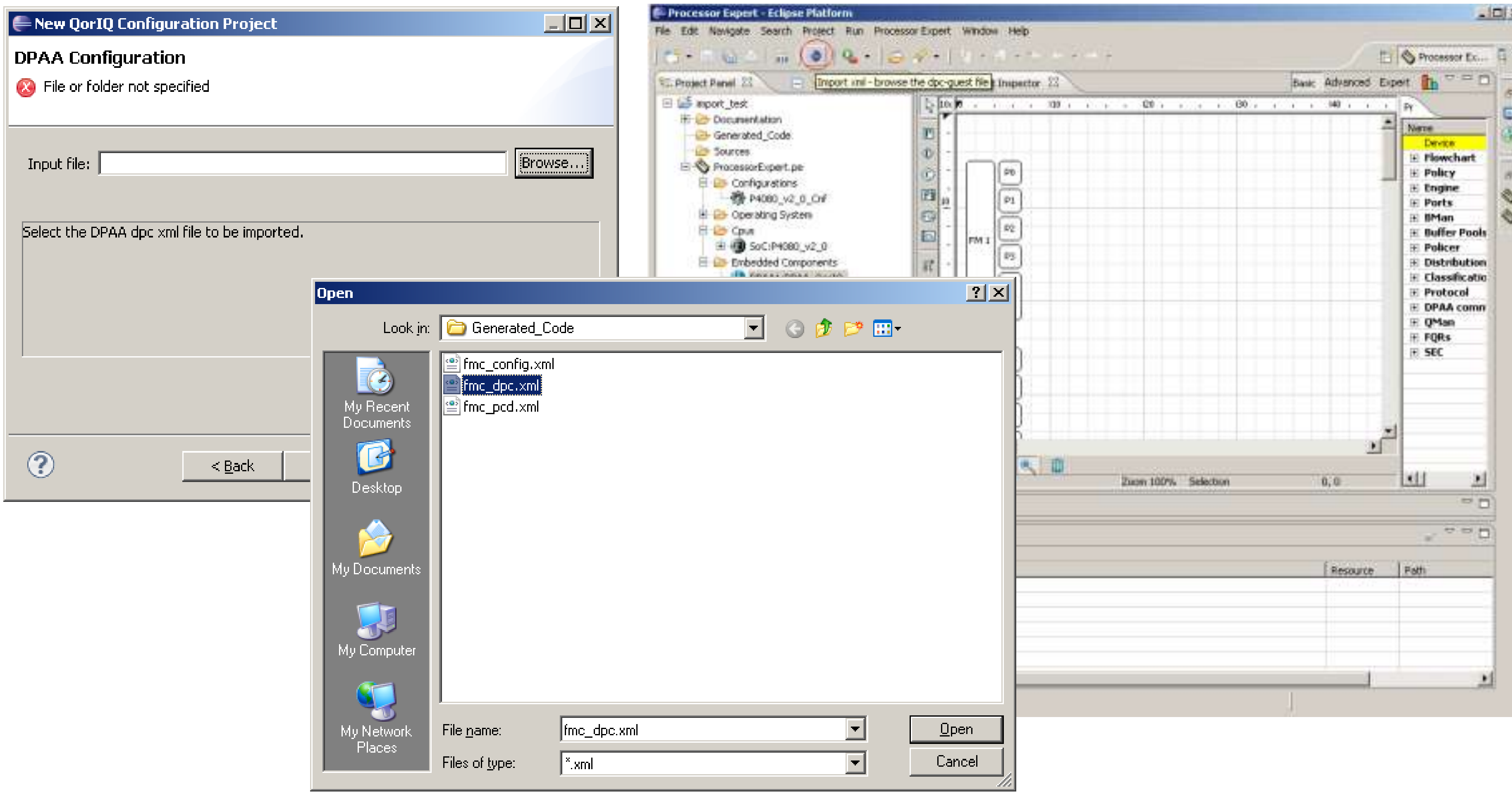
```xml
<?xml version="1.0"?>

<netpcd>
    <distribution name="Distribution1">
        <queue count="1" base="0x1"/>
        <action type="classification" name="Classification1"/>
        <protocols>
            <protocolref name="vlan"/>
        </protocols>
    </distribution>
    <distribution name="Distribution2">
        <queue count="2" base="0x2"/>
        <key>
            <fieldref name="ethernet.src"/>
            <fieldref name="llc_snap.type"/>
        </key>
        <protocols>
            <protocolref name="vlan"/>
        </protocols>
    </distribution>
```

freescale™

Import XMLs Feature

- **The xmls can be imported at project creation time or later after the project is created**

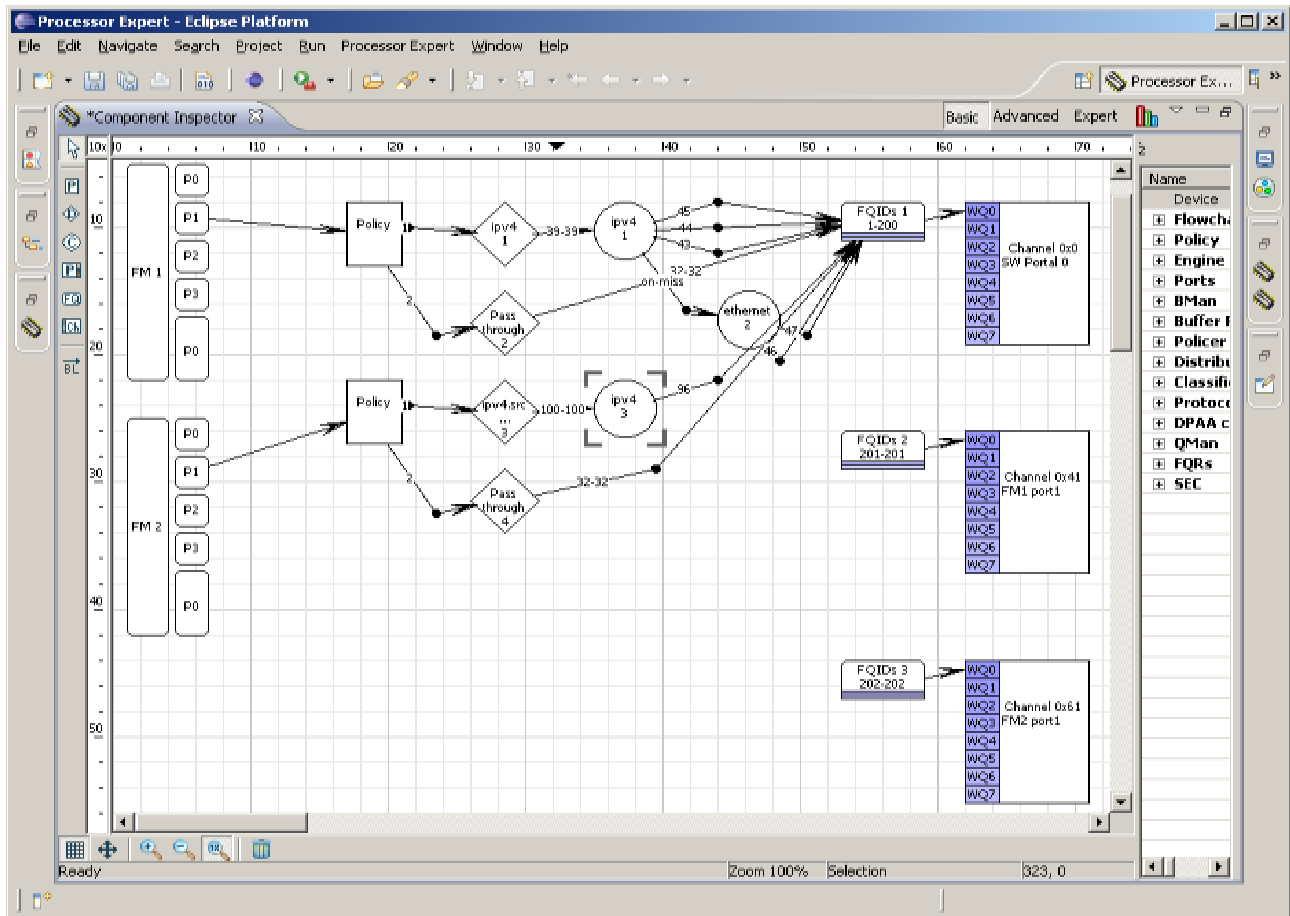


freescale™

Import XMLs Feature – Demo

- Importing files previously generated by the QCS tool won’t produce the same output
- The objects’ coordinates are not saved in the xmls and are calculated using a “placement algorithm” at the import time
- Import demo:

  Using the xmls generated by the hands-on scenario presented in the upper slides



freescale™

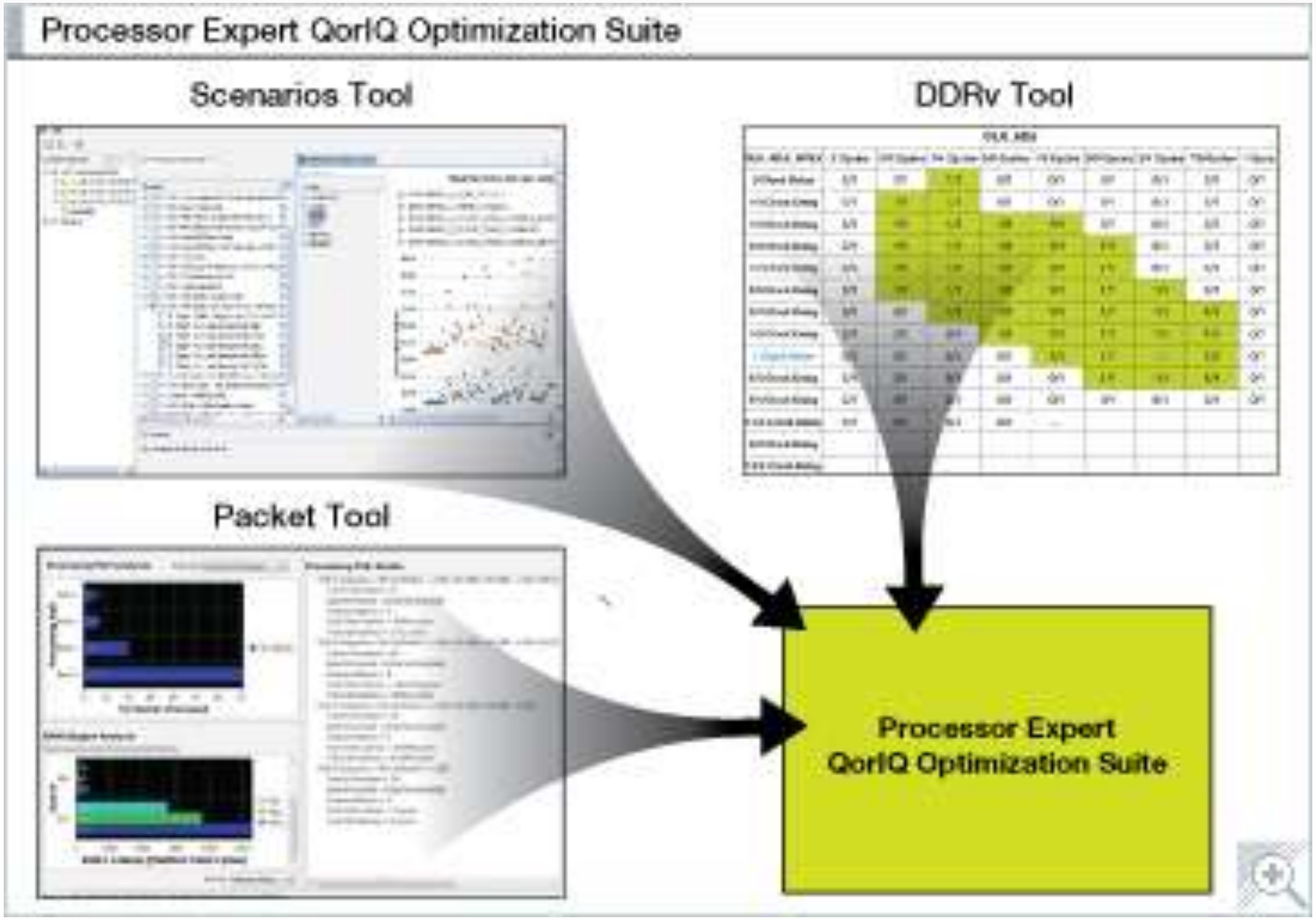
Processor Expert

QorIQ Optimization Suite





QorIQ Optimization Suite



What is the PEx Optimization Suite?

- A new generation of products aimed at allowing customers to solve systems and application performance problems in the QorIQ and Layerscape Family of devices.
  - Users can analyze their applications unencumbered by the complexity of the debug IP
  - Provides a simple, clear and concise way of configuring the QorIQ debug IP to solve performance problems.
  - Continues the usage of the proven Scenarios Concept providing customers with ‘recipes’ to analyze common and complex performance problems
  - Transfer Freescale’s knowledge to customers – Scenarios.
  - Supports bare metal and Linux applications. Special focus on Linux User space applications.

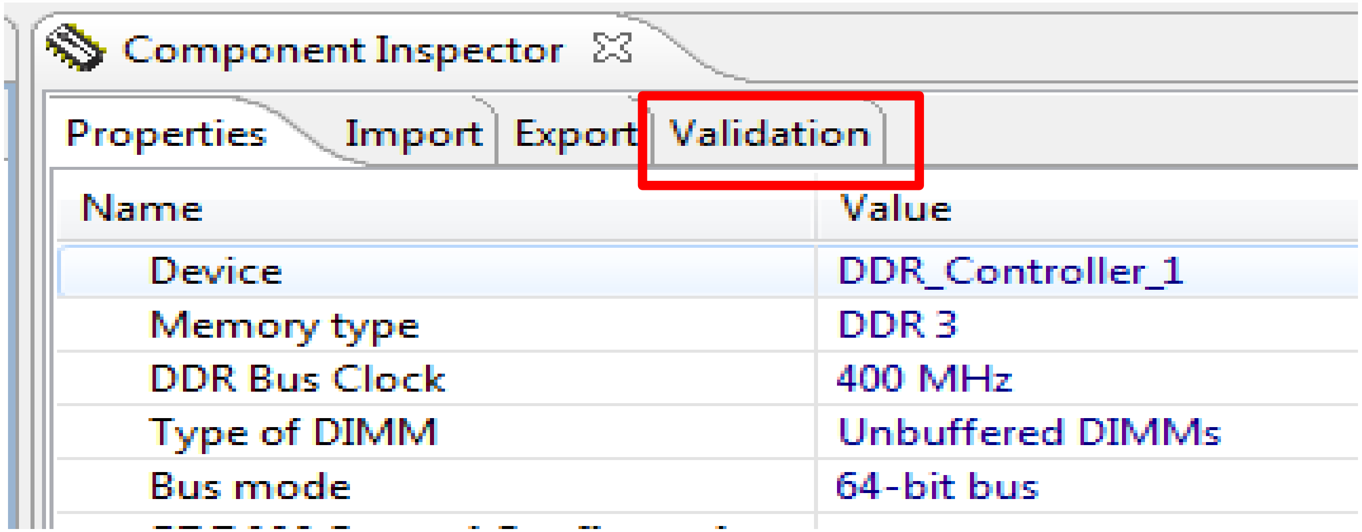
DDR validation tool

… extends QorIQ Configuration Suite





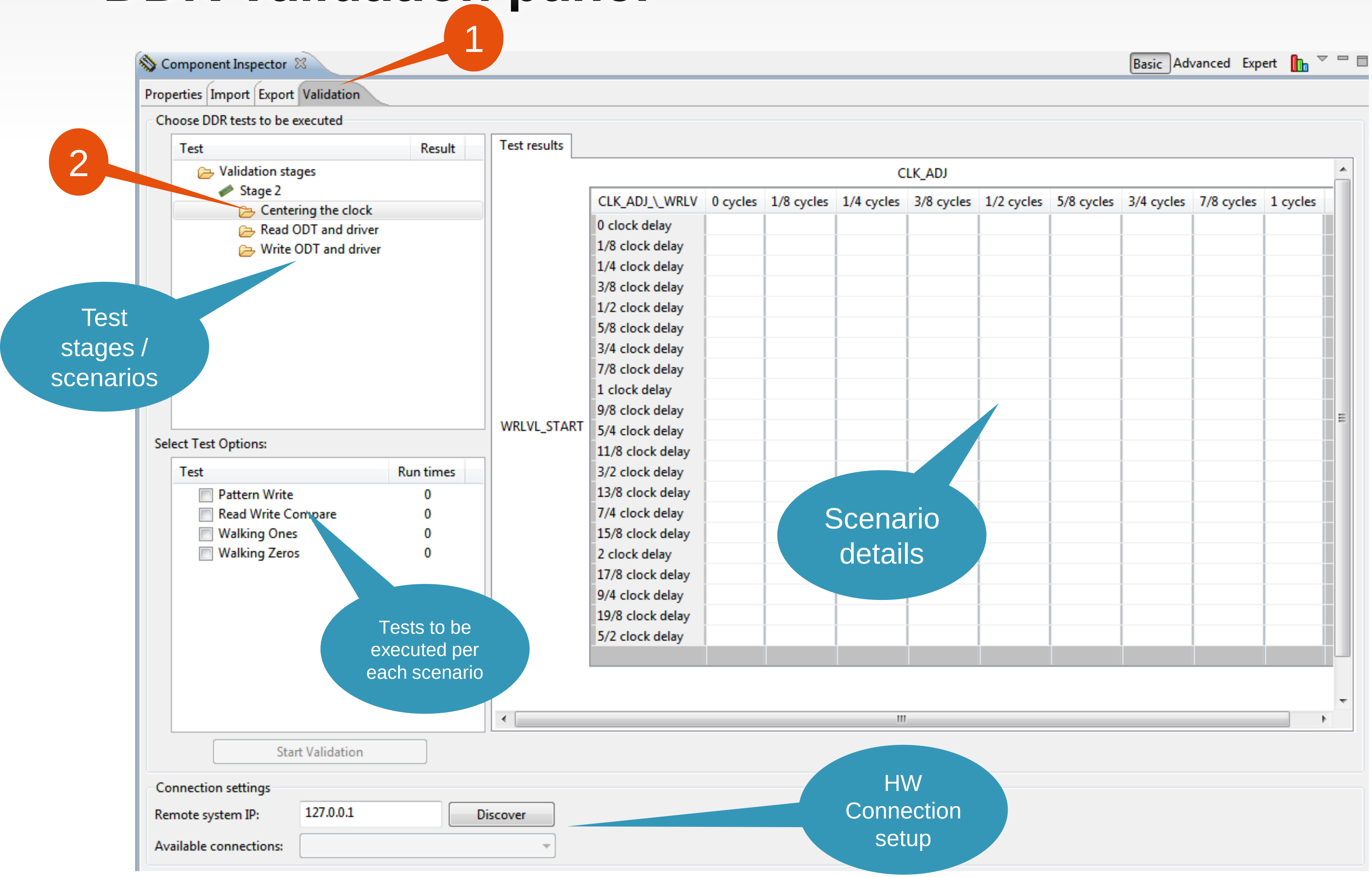
DDR Validation is a licensed product leveraging QCS



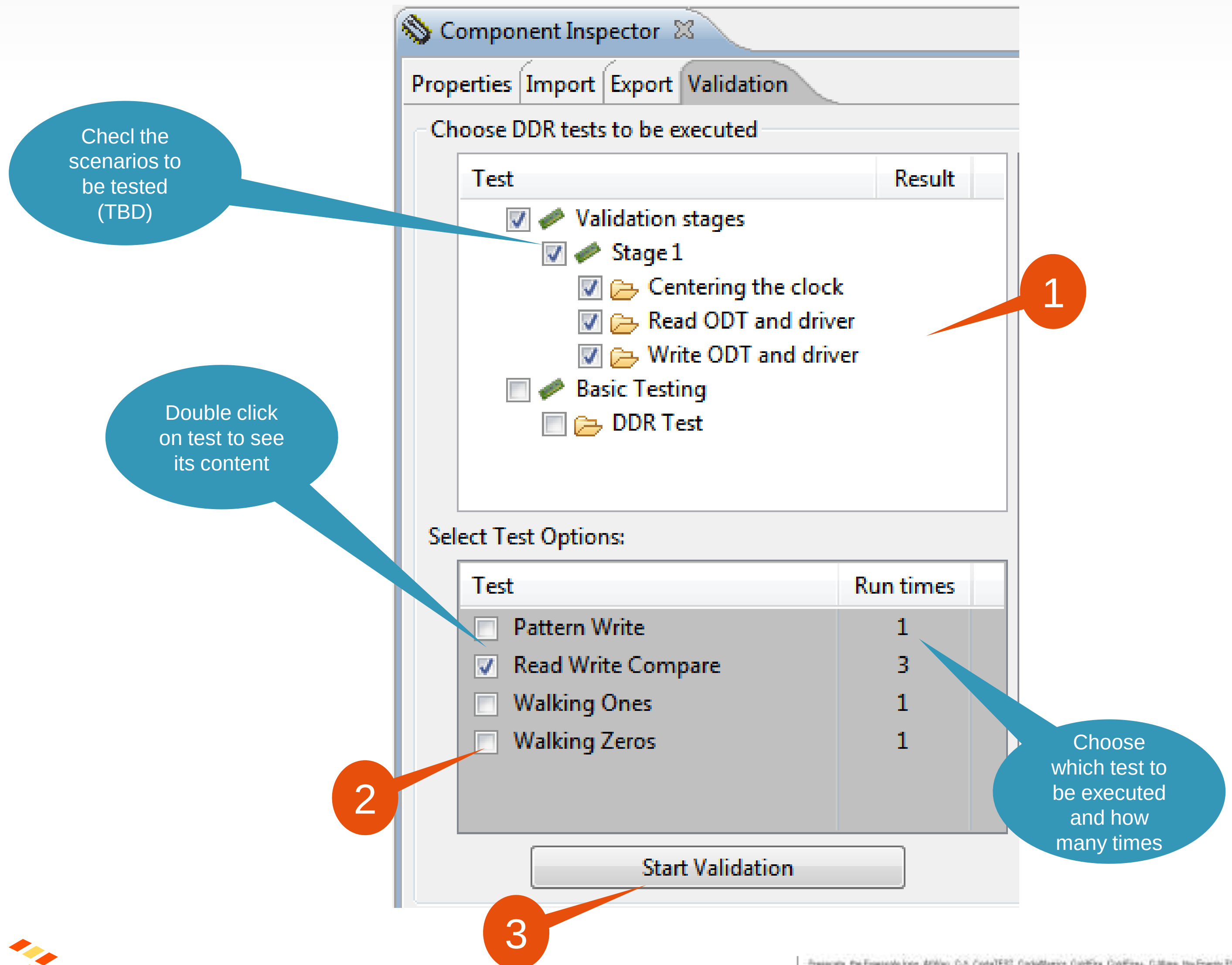
Pricing $995

License file:
<QCS Install directory>/eclipse/Optimization/license.dat

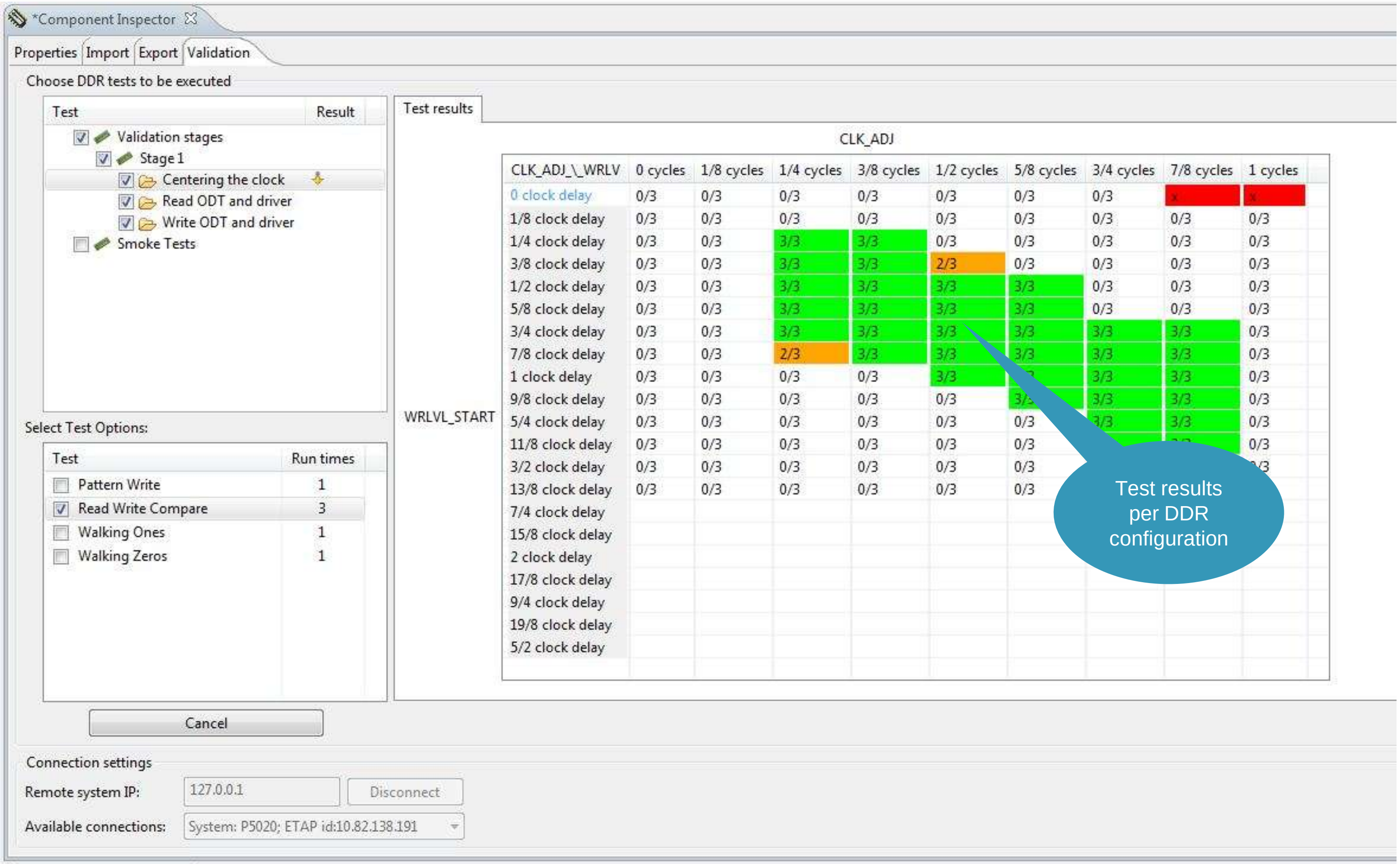
DDR Validation panel



Configure DDR tests to be run

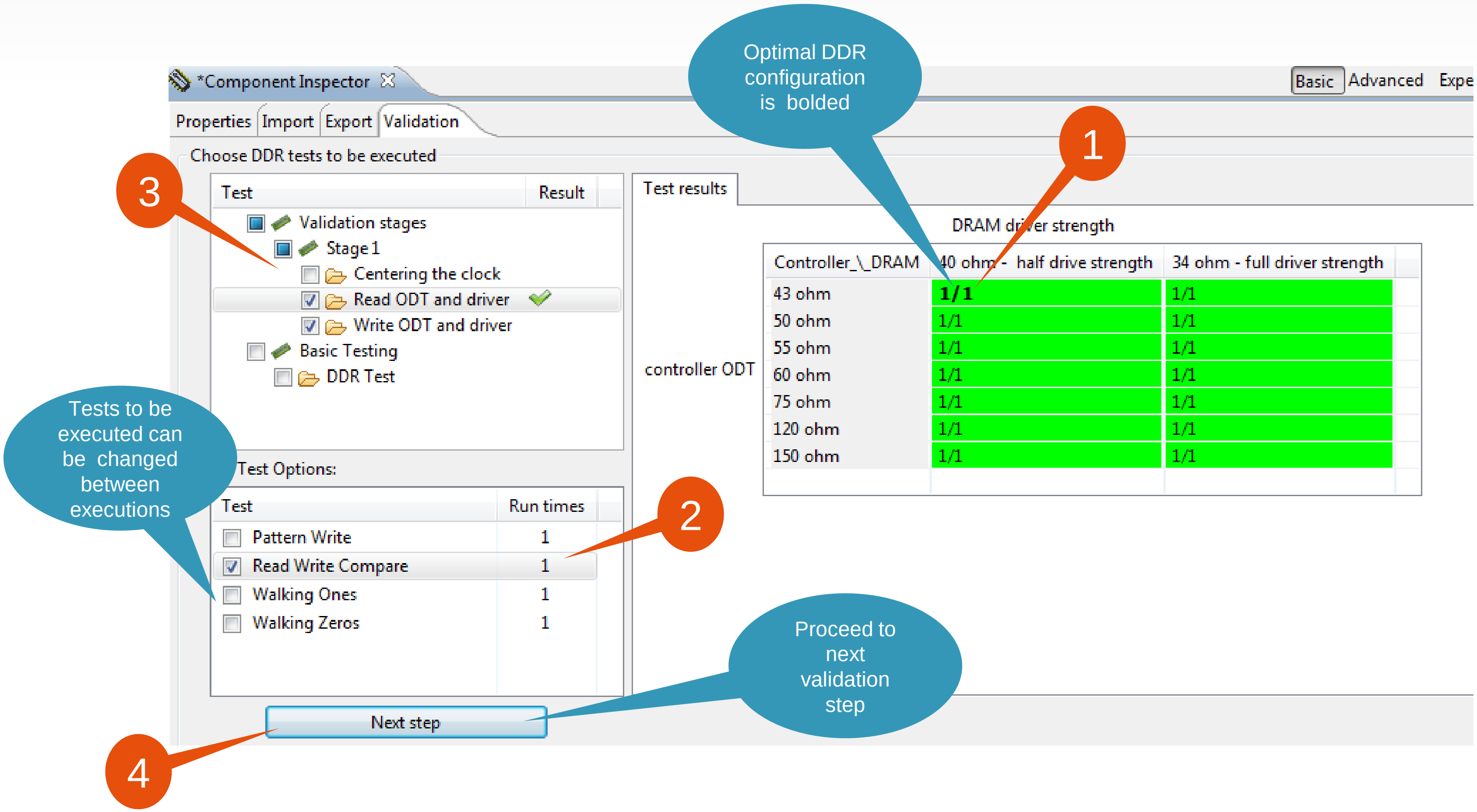


Observe DDR validation test results

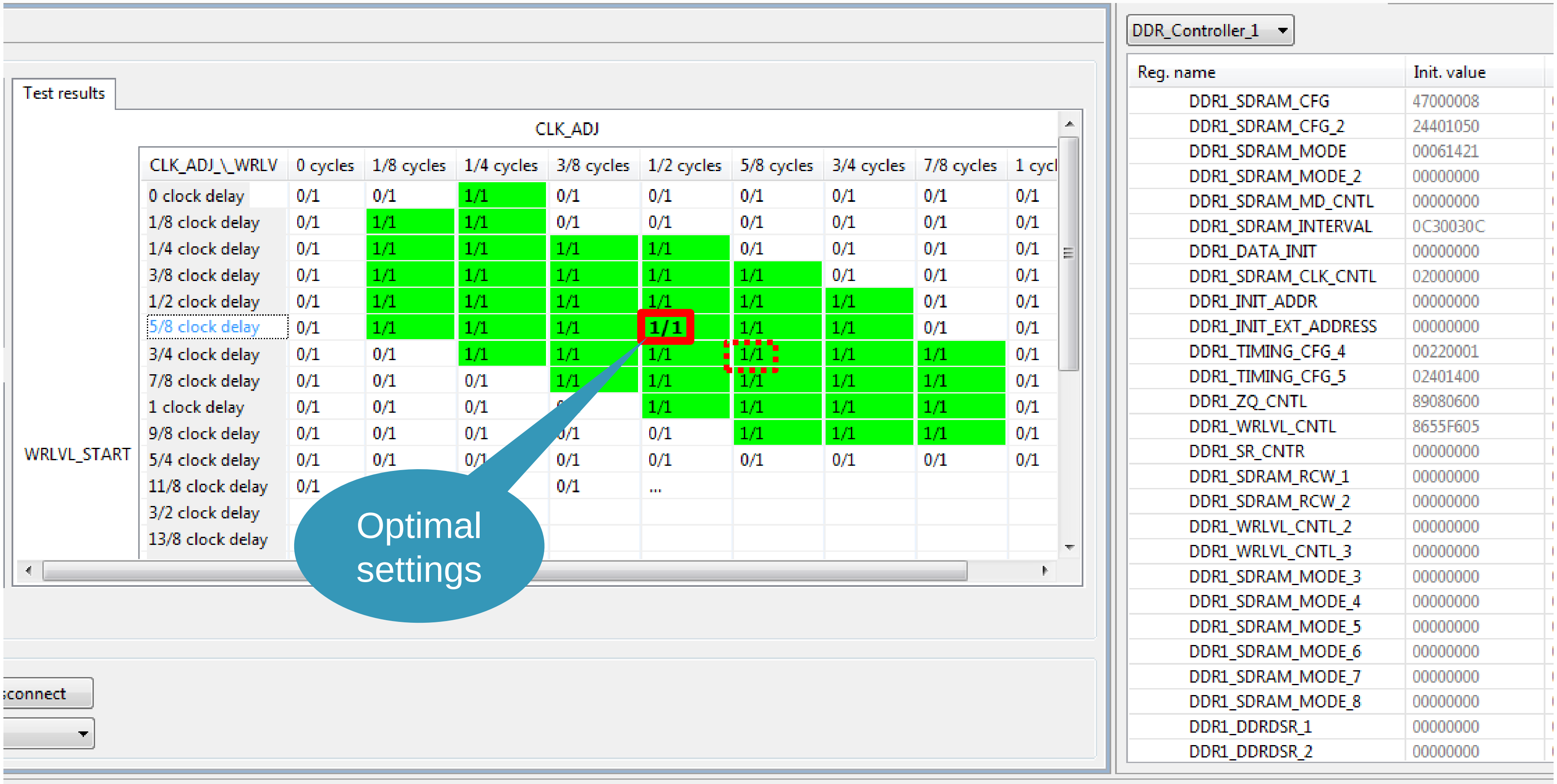


CLK_ADJ _WRLV	0 cycles	1/8 cycles	1/4 cycles	3/8 cycles	1/2 cycles	5/8 cycles	3/4 cycles	7/8 cycles	1 cycles
0 clock delay	0/3	0/3	0/3	0/3	0/3	0/3	0/3	x	x
1/8 clock delay	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3
1/4 clock delay	0/3	0/3	3/3	3/3	0/3	0/3	0/3	0/3	0/3
3/8 clock delay	0/3	0/3	3/3	3/3	2/3	0/3	0/3	0/3	0/3
1/2 clock delay	0/3	0/3	3/3	3/3	3/3	3/3	0/3	0/3	0/3
5/8 clock delay	0/3	0/3	3/3	3/3	3/3	3/3	0/3	0/3	0/3
3/4 clock delay	0/3	0/3	3/3	3/3	3/3	3/3	3/3	3/3	0/3
7/8 clock delay	0/3	0/3	2/3	3/3	3/3	3/3	3/3	3/3	0/3
1 clock delay	0/3	0/3	0/3	0/3	3/3	[illegible]	3/3	3/3	0/3
9/8 clock delay	0/3	0/3	0/3	0/3	0/3	[illegible]	3/3	3/3	0/3
5/4 clock delay	0/3	0/3	0/3	0/3	0/3	0/3	[illegible]	3/3	0/3
11/8 clock delay	0/3	0/3	0/3	0/3	0/3	0/3	[illegible]	[illegible]	0/3
3/2 clock delay	0/3	0/3	0/3	0/3	0/3	0/3	[illegible]	[illegible]	[illegible]
13/8 clock delay	0/3	0/3	0/3	0/3	0/3	0/3	[illegible]	[illegible]	[illegible]
7/4 clock delay									
15/8 clock delay									
2 clock delay									
17/8 clock delay									
9/4 clock delay									
19/8 clock delay									
5/2 clock delay									

Optimal DDR configuration (Read ODT)



P2020RDB-PCA DDRV optimized test results



CLK_ADJ__WRLV	0 cycles	1/8 cycles	1/4 cycles	3/8 cycles	1/2 cycles	5/8 cycles	3/4 cycles	7/8 cycles	1 cycl
0 clock delay	0/1	0/1	1/1	0/1	0/1	0/1	0/1	0/1	0/1
1/8 clock delay	0/1	1/1	1/1	0/1	0/1	0/1	0/1	0/1	0/1
1/4 clock delay	0/1	1/1	1/1	1/1	1/1	0/1	0/1	0/1	0/1
3/8 clock delay	0/1	1/1	1/1	1/1	1/1	1/1	0/1	0/1	0/1
1/2 clock delay	0/1	1/1	1/1	1/1	1/1	1/1	1/1	0/1	0/1
5/8 clock delay	0/1	1/1	1/1	1/1	**1/1**	1/1	1/1	0/1	0/1
3/4 clock delay	0/1	0/1	1/1	1/1	1/1	1/1	1/1	1/1	0/1
7/8 clock delay	0/1	0/1	0/1	1/1	1/1	1/1	1/1	1/1	0/1
1 clock delay	0/1	0/1	0/1		1/1	1/1	1/1	1/1	0/1
9/8 clock delay	0/1	0/1	0/1	0/1	0/1	1/1	1/1	1/1	0/1
5/4 clock delay	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
11/8 clock delay	0/1			0/1	...				
3/2 clock delay									
13/8 clock delay									

Reg. name	Init. value
DDR1_SDRAM_CFG	47000008
DDR1_SDRAM_CFG_2	24401050
DDR1_SDRAM_MODE	00061421
DDR1_SDRAM_MODE_2	00000000
DDR1_SDRAM_MD_CNTL	00000000
DDR1_SDRAM_INTERVAL	0C30030C
DDR1_DATA_INIT	00000000
DDR1_SDRAM_CLK_CNTL	02000000
DDR1_INIT_ADDR	00000000
DDR1_INIT_EXT_ADDRESS	00000000
DDR1_TIMING_CFG_4	00220001
DDR1_TIMING_CFG_5	02401400
DDR1_ZQ_CNTL	89080600
DDR1_WRLVL_CNTL	8655F605
DDR1_SR_CNTR	00000000
DDR1_SDRAM_RCW_1	00000000
DDR1_SDRAM_RCW_2	00000000
DDR1_WRLVL_CNTL_2	00000000
DDR1_WRLVL_CNTL_3	00000000
DDR1_SDRAM_MODE_3	00000000
DDR1_SDRAM_MODE_4	00000000
DDR1_SDRAM_MODE_5	00000000
DDR1_SDRAM_MODE_6	00000000
DDR1_SDRAM_MODE_7	00000000
DDR1_SDRAM_MODE_8	00000000
DDR1_DDRDSR_1	00000000
DDR1_DDRDSR_2	00000000

freescale™

P2020RDB-PCA: Compare optimal DDR configuration with uboot chosen values

Read DDR configuration from uboot

=> md ffe02000

ffe02000: 0000003f 00000000 00000000 00000000 .

ffe02080: 80014202 00000000 00000000 00000000

ffe02100: 00030000 00110104 6f6b8846 0fa8c8cc ...

ffe02110: c7000008 24401040 00441421 00000000 ....$@.@.D.!....

ffe02120: 00000000 0c300100 deadbeef 00000000 ..

ffe02130: 03000000 00000000 00000000 00000000

ffe02160: 00220001 02401400 00000000 00000000

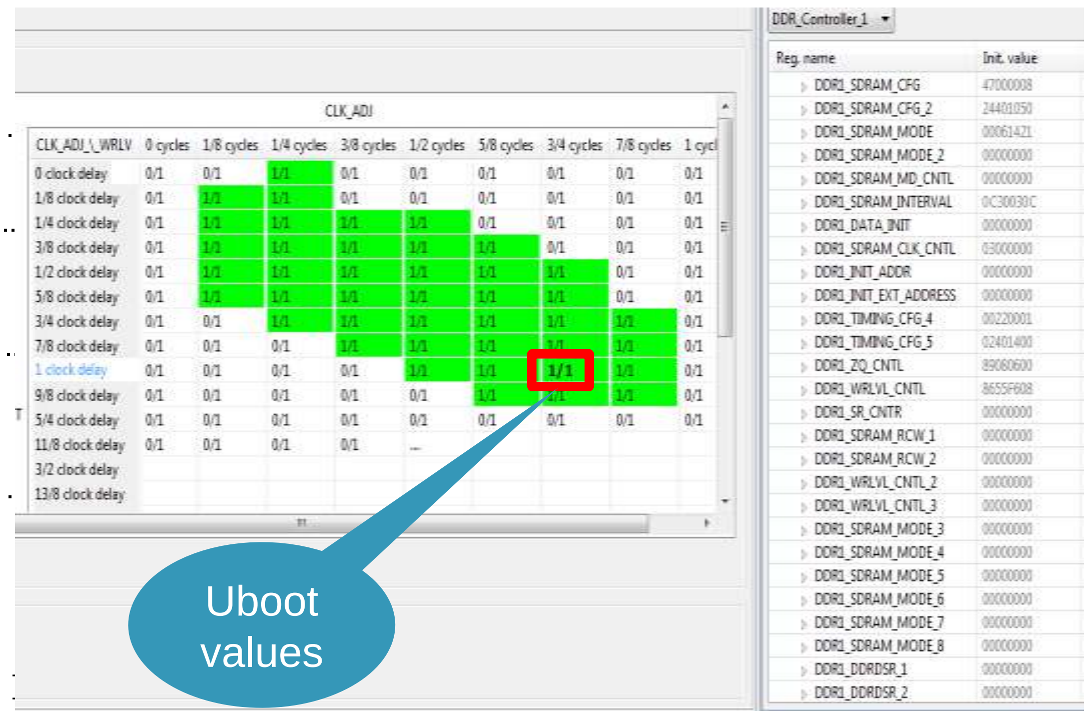
ffe02170: 89080600 8675f608 00000000 00000000 .

=> md ffe02b00

ffe02b00: 00000000 00000000 00000000 00000000

ffe02b10: 00000000 00000000 00000000 00000000

ffe02b20: 5dc07777 77000000 00000000 00000000



DDR Validation Tool Availability

- **You need QCS 2.2.1 or later installed. Get it from www.freescale.com/qcs**
- **Use Eclipse "Add new software.." Eclipse updater capability to install DDRV on top of QCS. Use as an update link:** http://freescale.com/lgfiles/updates/Eclipse/Helios36/DDRValidation for installation over QCS installed over Eclipse 3.6 or CW PA 10
- http://freescale.com/lgfiles/updates/Eclipse/Indigo37/DDRValidation for installation over QCS installed over Eclipse 3.7
- **DDRV is a licensed product.**

QorIQ Scenario Tool

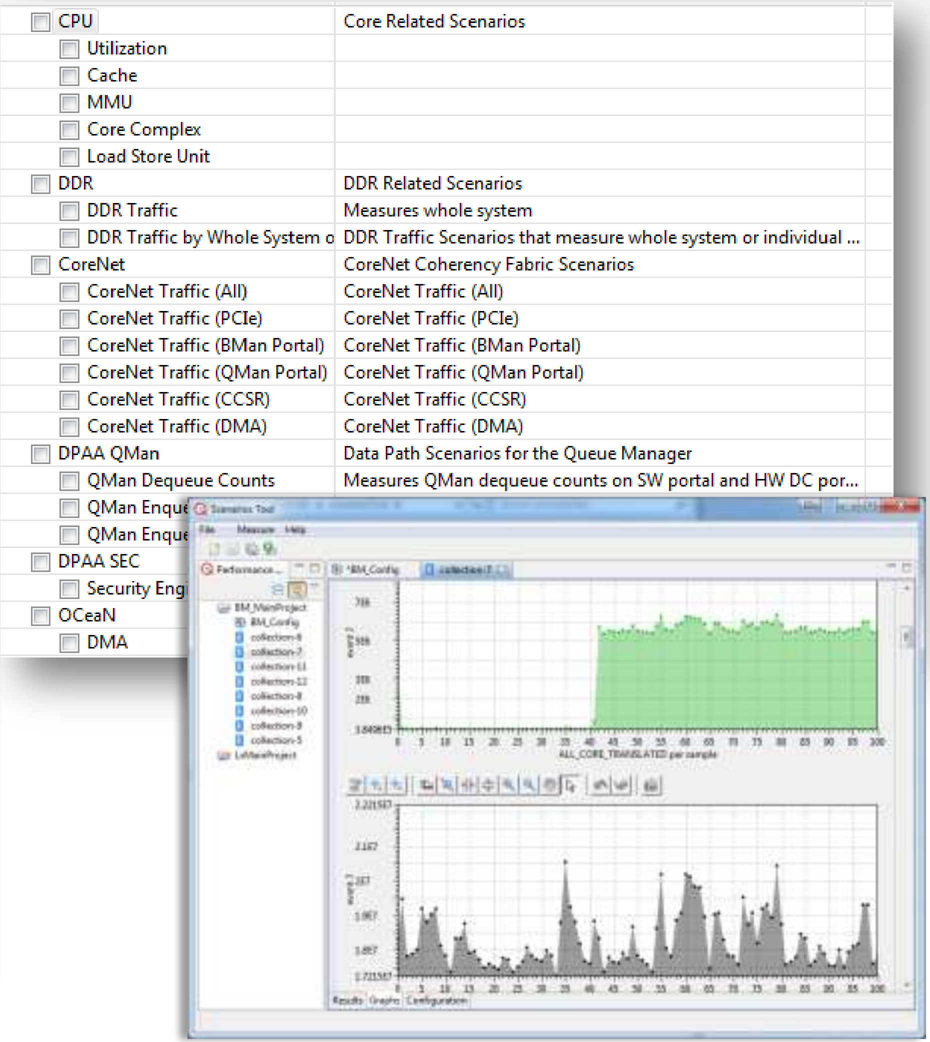
a tool in QorIQ Optimization Suite





Optimization Suite – Scenarios Tool

- **Customer Benefit:**
  - System Optimization – Cores and SoC
  - Complexity Abstraction and ease of use.
  - Streamlined to solve several performance issues
  - Deliver FSL expertise to users
  - Probe-less, field based usage.
- **Target areas:**
  - Select QorIQ devices (P3 – T4/B4) and future Layerscape devices.
  - Linux Systems (focus), but also supports bare metal
  - Performance Analysis including visualization

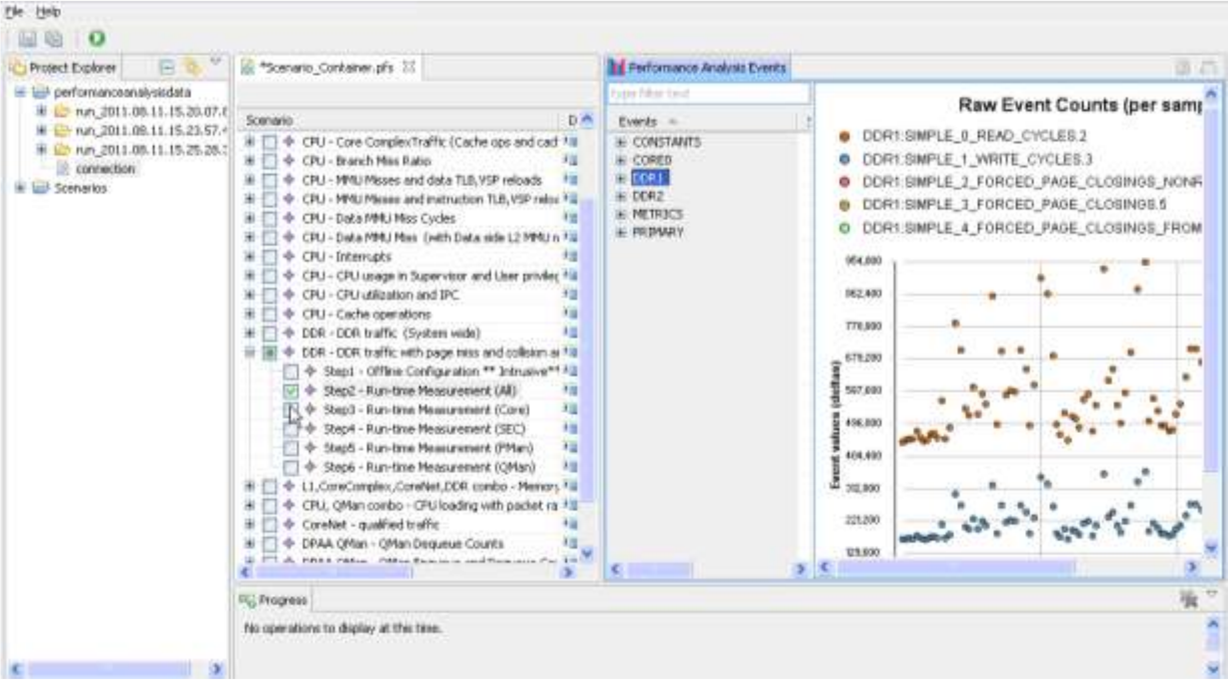


Optimized workflow for efficiently narrowing down performance issues anywhere on the system

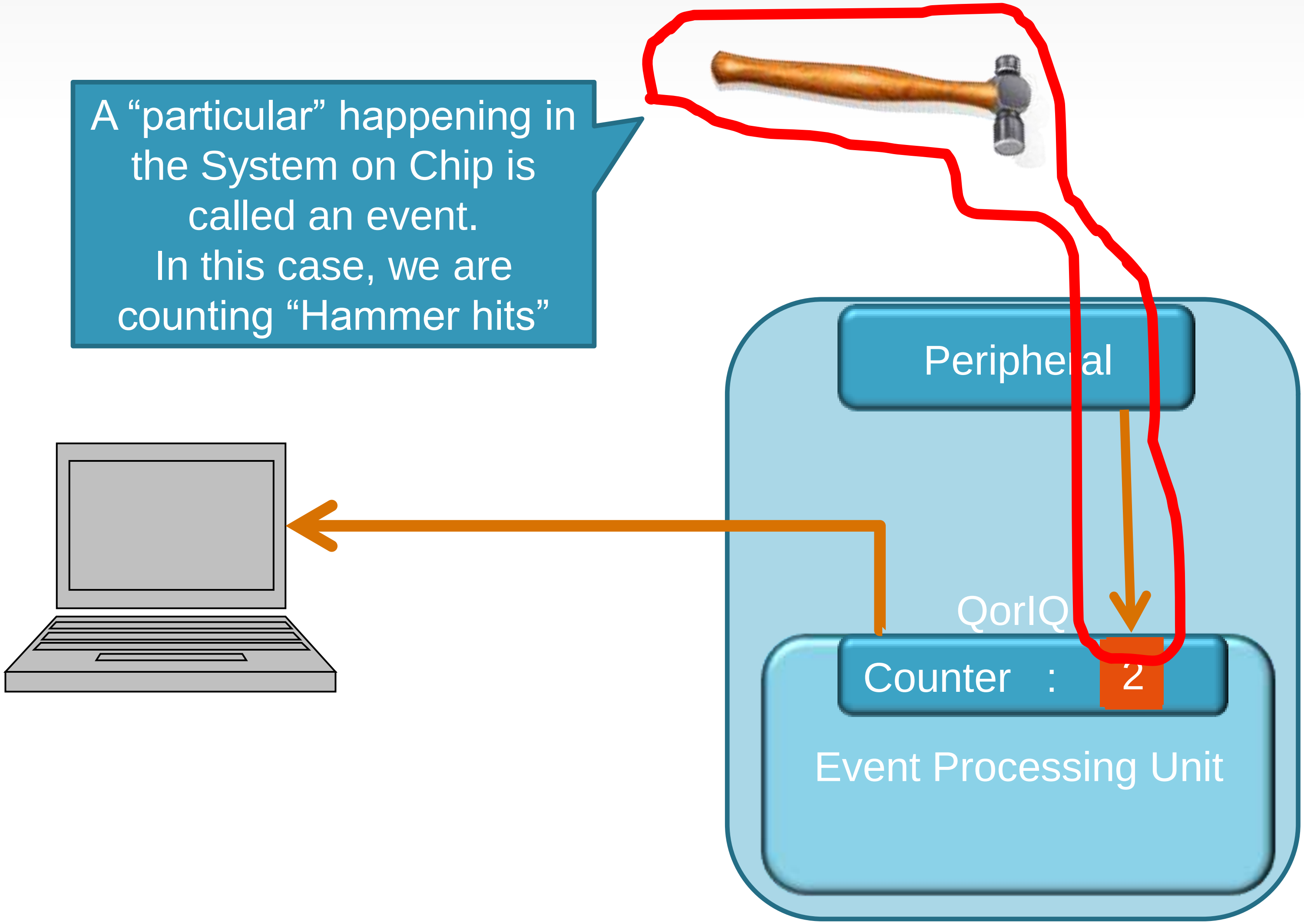
freescale™

What is Scenarios Tool?

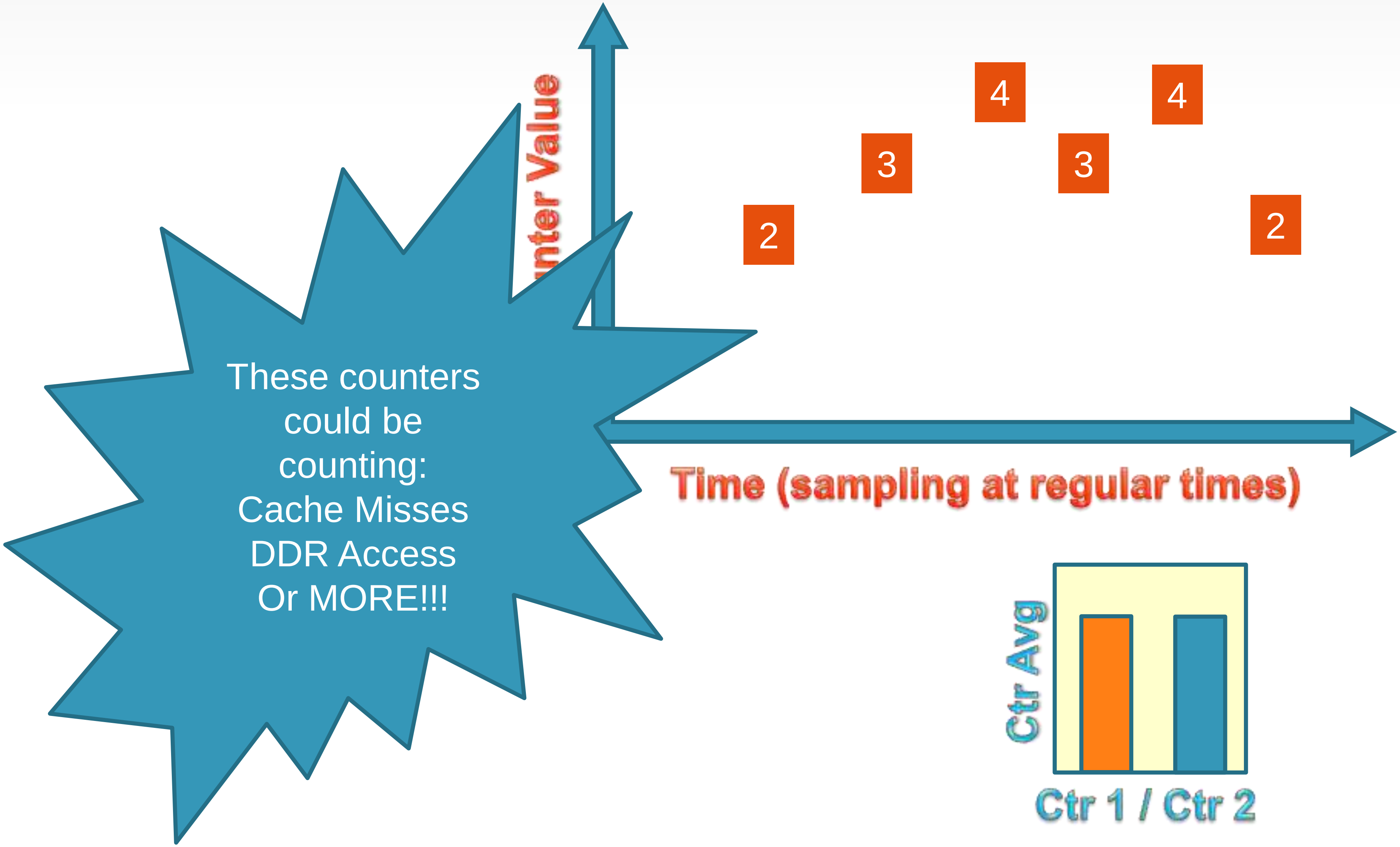
– **Visually identify the system level problem areas in seconds**.



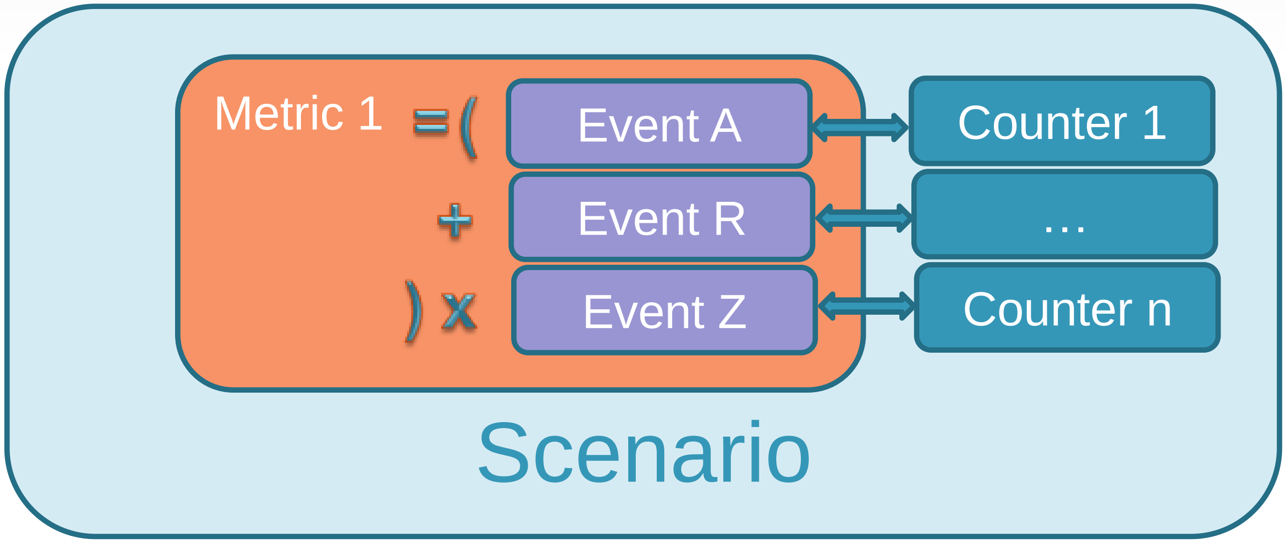
How "counters" work



Making Sense of Counter Data



What is a scenario?



A scenario is a CONTAINER of :

- required counters
- key events for a measurement
- counter<->event connections
- metrics
    - a metric is a math equation of captured event-counts

A GOOD scenario would count and combine (using metrics) all of the information required to measure something useful, such as "cache misses" or "buffer overflows"

freescale™

Here Scenarios Tool measures cache accesses

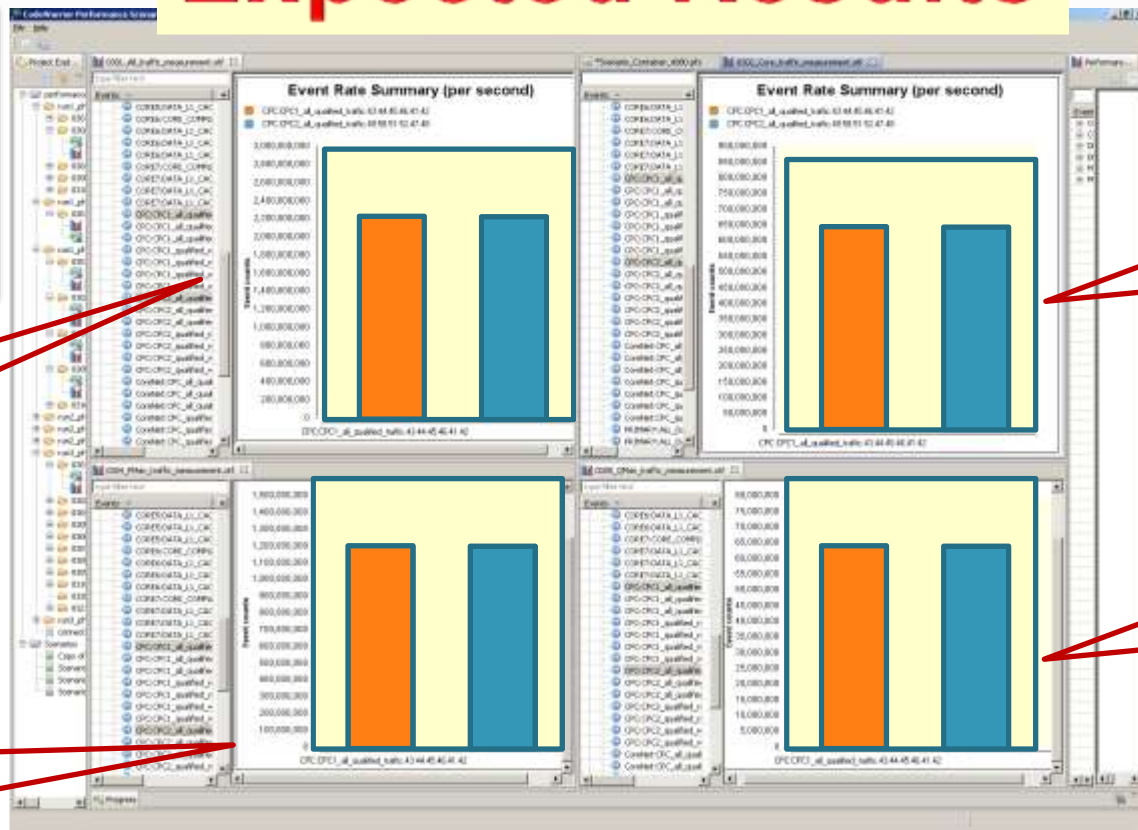
Expected Results

Selected scenario captures overall traffic as well as individual subsystem traffic including traffic from cores, Fman and Qman

Total traffic to CPC1 and CPC2

Unbalance between platform caches

Fman CPC2 Access



CPU Accesses

Qman CPC2 Access

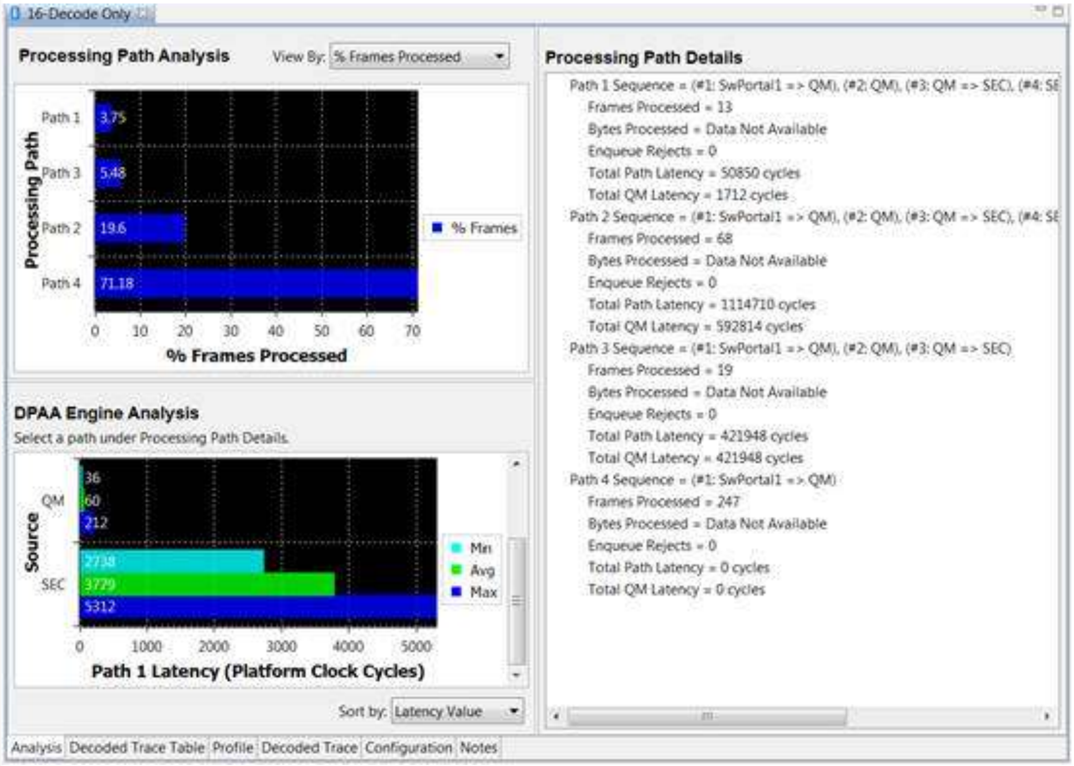
- Imbalance can be seen by analyzing these subsystems:
  - Majority of CPC1 accesses are made by the CPU (top right)
  - Majority of CPC2 access are from Fman and Qman
  - Remainder of the traffic is due to PCI
- LACK OF BALANCED ACCESSES suggests DDR interleaving is not configured properly.
- Fixing the configuration provides a performance boost.

QorIQ Packet Analysis Tool



Packet Analysis Tool

- **Customer Benefits**
  - Complexity abstraction and ease of use
  - Enables key use cases:
    - Packet-Oriented System Level Performance Analysis
    - SoC Data Plane Configuration Debug
    - Packet Processing Latency Analysis
    - Packet Processing Critical Resource Monitoring
- **Target areas:**
  - SoC debug/analysis feature enablement
  - Linux Systems
  - Analysis data interpretation and visualization
- **Users**
  - External Customers
  - Freescale internal developers



Main Use Cases

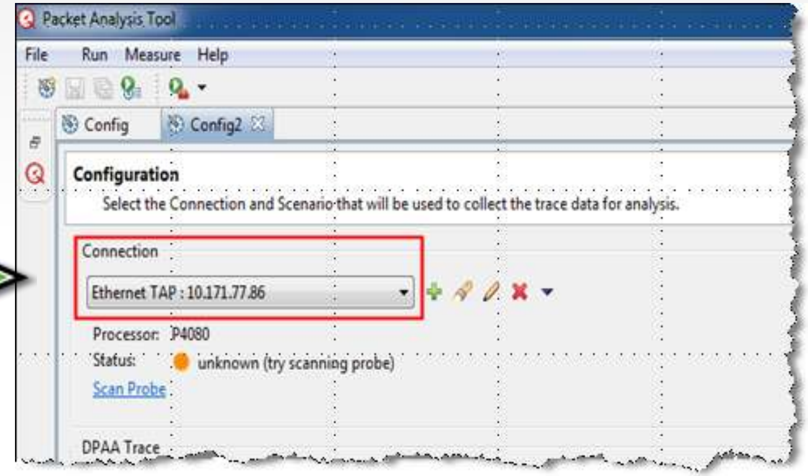
Packet Tracing	Shows which parts of the system process the frames. For example, use this to verify that the frame flow is what you expect.
Lost Packet Analysis	Understand why the frames become “lost” in the system. For example, use this to check how the FM PCD changes affect where frames are sent.
Latency Analysis	Precisely measure the time spent processing frames at various points in the system.
Packet Sequence Analysis	See how an entire sequence of frames was processed. For example, use this to measure the performance of the SEC.
QM Performance Analysis	Use QM profile data to measure the performance of the system, at “data-flow” level.

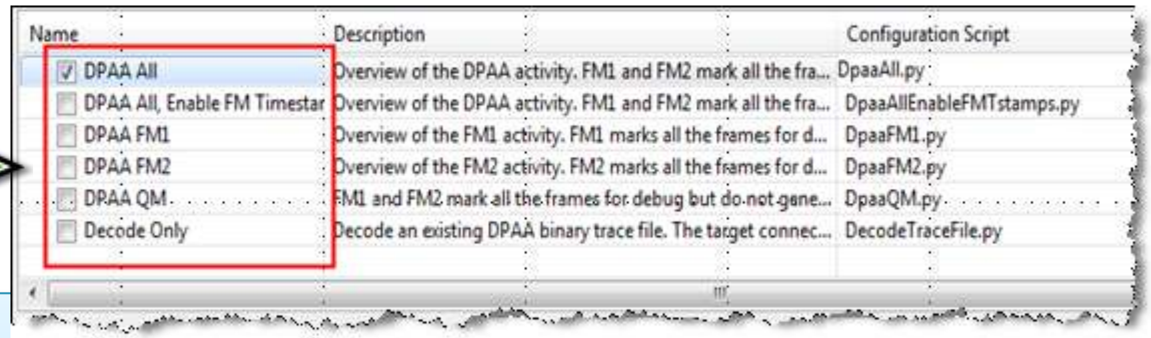
QorIQ DataPath Trace

- Visibility into FM and QM activities via Nexus Trace
- **Trace data**
  - Can be collected from a running system
    - Without interrupting it and
    - Without affecting its performance
  - Can be collected to on-board trace buffer
  - Is timestamped so it can be used to precisely measure the timings
- **FM trace**
  - Optionally output by each FM engine: BMI, KeyGen, Parser etc
  - Timestamped internally by the FM clock
  - The trace data contains: FD, FM port number, NIA etc
- **QM trace**
  - Optionally output by each QM enqueue and dequeue point
  - The trace data contains: FQID, channel, frame address, frame length, enqueue/dequeue flag, portal type and number etc
- **Traced frames**
  - Only the tagged frames are traced. “Tagged” = FD[DD] bits set.
  - Rx flow – the frames tagged by the FM, as configured by the Packet Analysis Tool
  - Other flows – the frames tagged by the instrumented software running on the cores

Configuring the Packet Tools

Target connection
TAP configuration and system type (e.g. P5020)





Scenarios

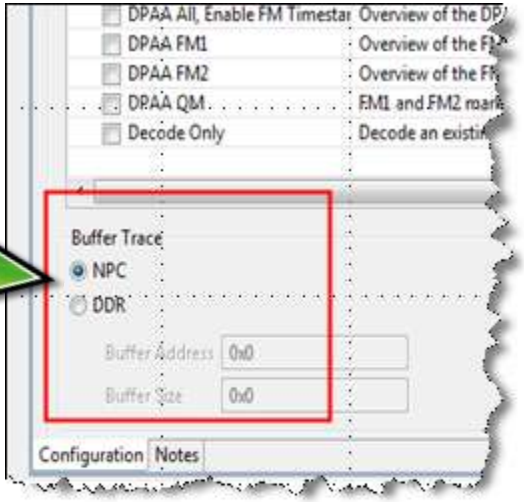
- Python scripts containing predefined trace setting configurations
- The scripts use the tool’s API for configuring the DPAA trace settings
- The tool ships with scenarios for tracing all frames in FM1, FM2 and QM, for tracing frames only in QM, only in FM1 etc.
- Also available: a scenario for analyzing trace from a file
- The users can define their own scenarios

freescale™

Configuring the Packet Tools

Trace buffer

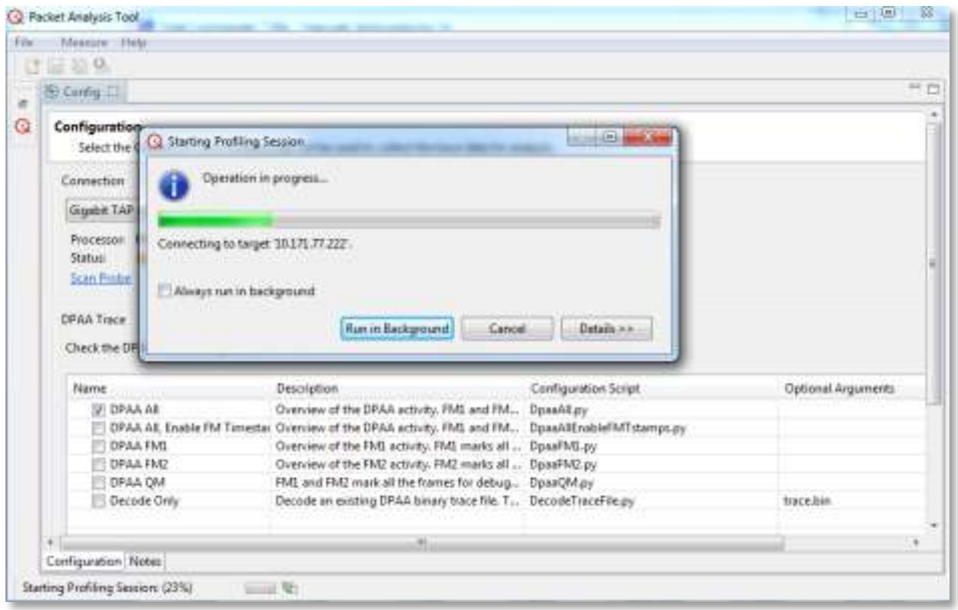
- NPC (Nexus Port Controller) – 16kB buffer for non-intrusive trace collection
- DDR – trace buffer located to the system's external memory, large capacity



- DDR Intrusiveness depends on the system's load and on the amount of trace data
- Use Uboot to reserve memory for it (see the user manual)

Data Collection

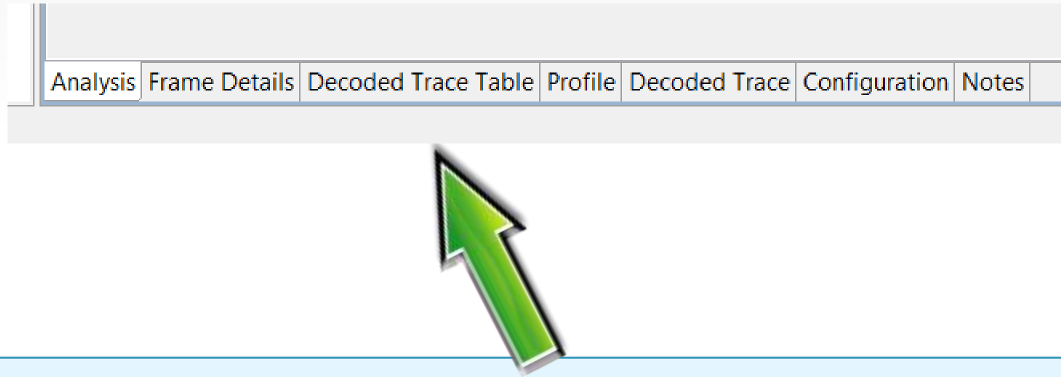
- After defining the configuration, select Measure/Launch:



- The results will appear in the Results folder of the project folder
- Results are collected as part of the project and will appear (if the tool layout uses the default view) on the left side of the tool

freescale™

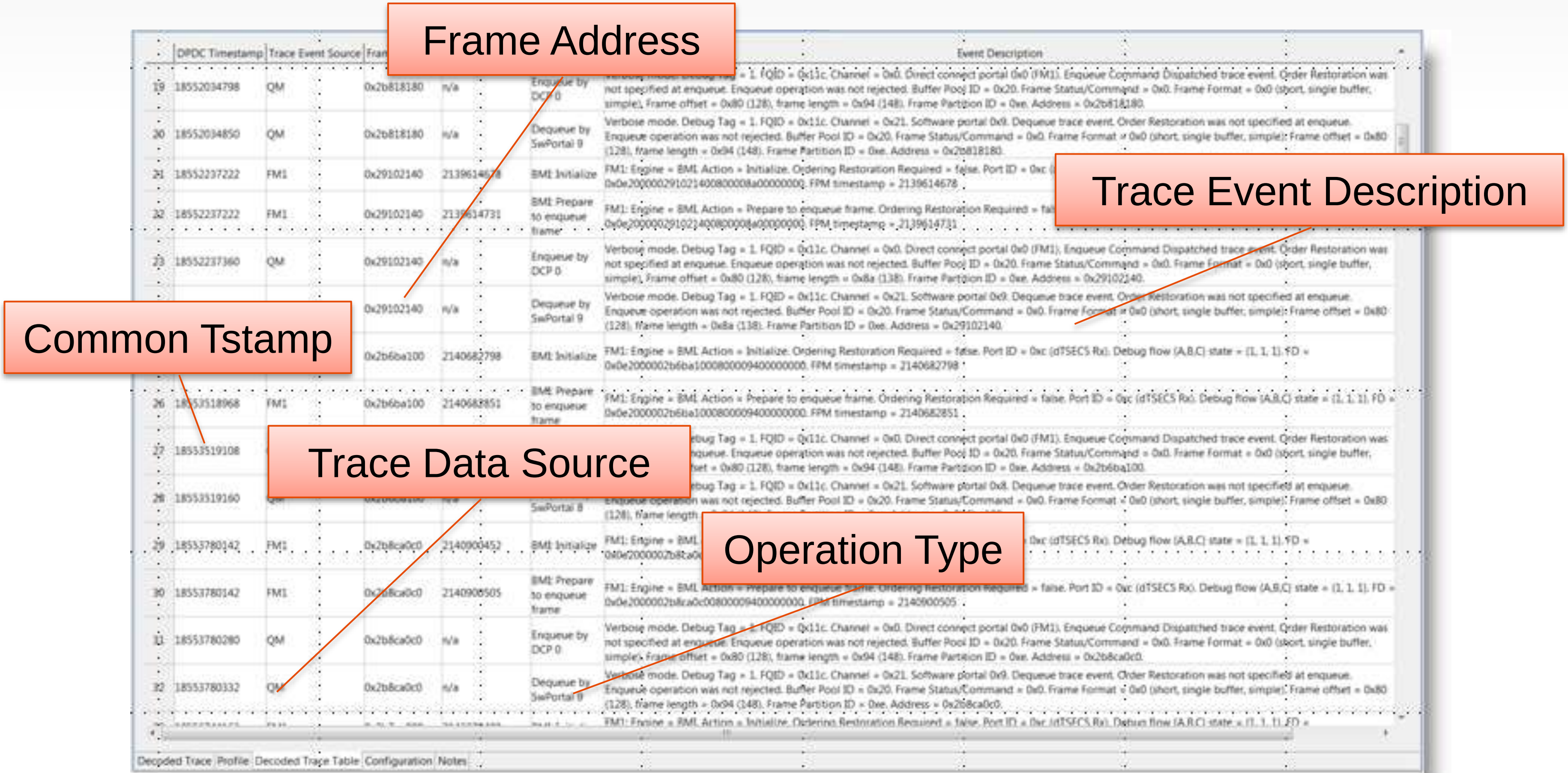
Data Display



Seven tabs are available at the bottom of the table view:

- **Analysis** – system level analysis data, based on the QM trace
- **Frame Details** – details for each traced frame
- **Decoded Trace Table** – a table view showing the trace data; the key attributes of the trace data, such as frame address, timestamp etc are displayed using separate columns
- **Profile** – a text view showing a trace based QM profile
- **Decoded Trace** – a text view showing the decoded trace data
- **Configuration** – a read-only display of the data that was used to collect the trace data
- **Notes** – displays and also allows entering new annotations for the analysis data

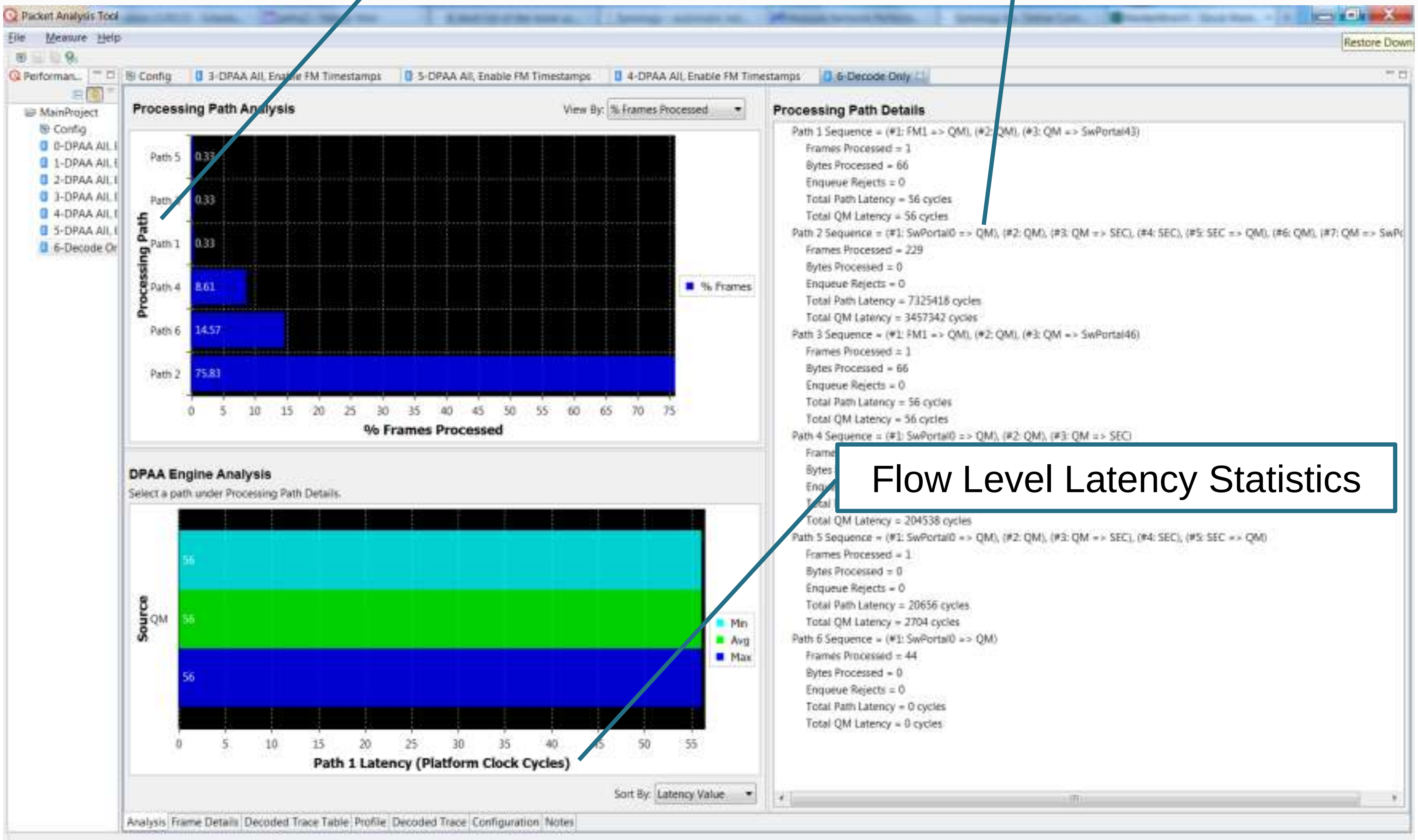
Data Display



- The data is automatically displayed after collection
- To see previously collected data, double-click on the project you want

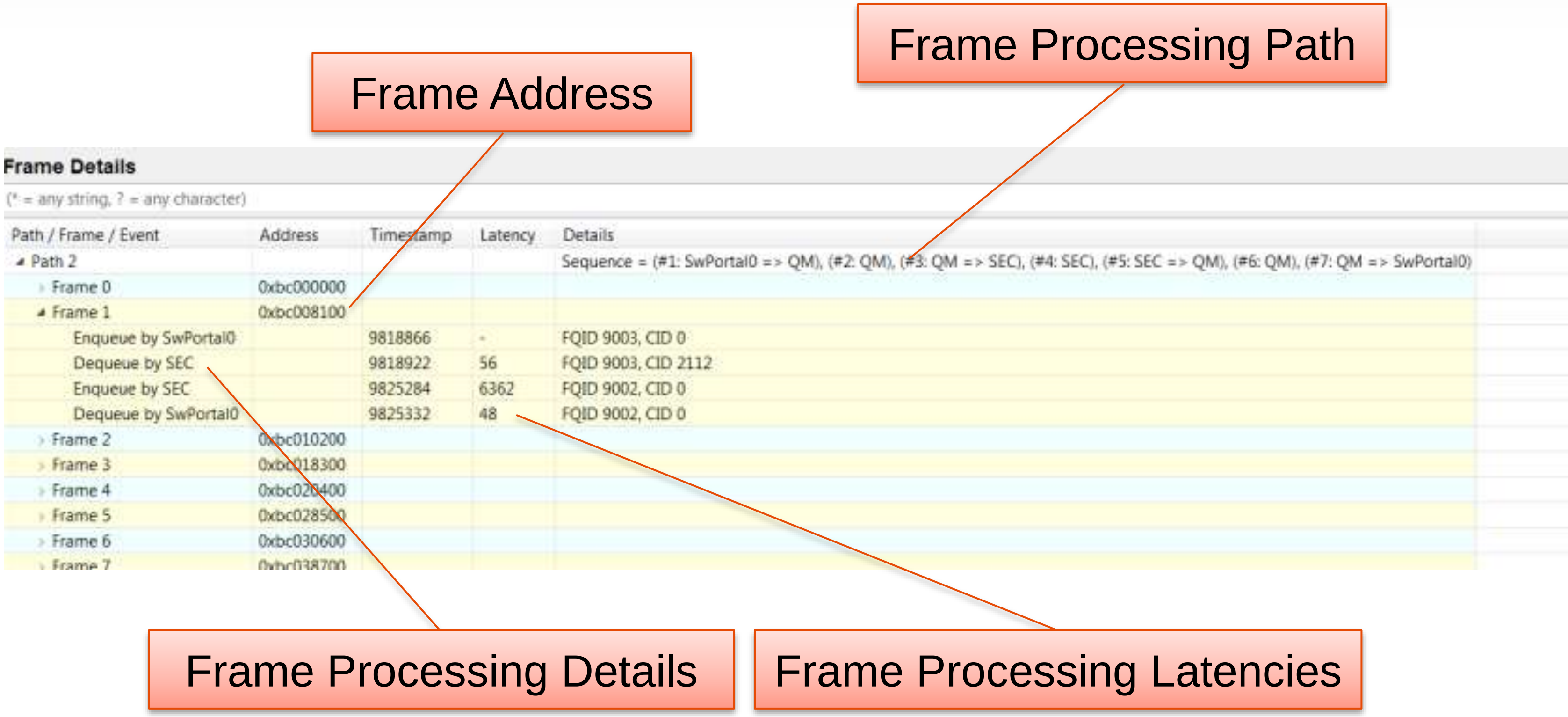
Data Flows From Trace Data

Frame Processing Stages



Flow Level Latency Statistics

Data Display



Frame Details

(* = any string, ? = any character)

Path / Frame / Event	Address	Timestamp	Latency	Details
Path 2				Sequence = (#1: SwPortal0 => QM), (#2: QM), (#3: QM => SEC), (#4: SEC), (#5: SEC => QM), (#6: QM), (#7: QM => SwPortal0)
Frame 0	0xbc000000			
Frame 1	0xbc008100			
Enqueue by SwPortal0		9818866	-	FQID 9003, CID 0
Dequeue by SEC		9818922	56	FQID 9003, CID 2112
Enqueue by SEC		9825284	6362	FQID 9002, CID 0
Dequeue by SwPortal0		9825332	48	FQID 9002, CID 0
Frame 2	0xbc010200			
Frame 3	0xbc018300			
Frame 4	0xbc020400			
Frame 5	0xbc028500			
Frame 6	0xbc030600			
Frame 7	0xbc038700			

freescale™

Predefined configurations

Scenario Name	Description
DPAA All	Measures an overview of all of the DPAA activity
DPAA All, Enable FM Timestamps	Measures an overview of all of the DPAA Activity
DPAA FM1	Overview of FM1 activity, FM1 marks the frames for debug and generates trace.
DPAA FM2	Overview of FM2 activity, FM2marks the frames for debug and generates trace.
DPAA QM	FM1 and FM2 mark the frames for debug but don’t generate trace. Only QM generates Trace
Decode Only	Decodes and existing DPAA Binary Trace File

Summary

Summary

You should now

- Be familiar with the QorIQ Configuration Suite basics
  - v1 supports PBL & DDR configuration (available since June 2011)
  - v2 adds Device Tree and DPAA Graphing tools (available since Nov 2011)
  - v2.2 adds Bootrom tool & all QorIQ NPI (available since May 2012)
- Be familiar with the QorIQ Optimization Suite basics
  - DDRv for on-chip validation of DDR controller configuration
  - Scenarios tool for capturing running system results for various characteristics
  - Packet tool for visualizing the flow of packets within a QorIQ device

- **Solution/Strategy**
  - **Extensible suite of tools to solve these problems**
  - Consolidate into a common tools framework (Processor Expert)
  - Provide new device support aligned with silicon roadmap
  - Add more configuration tools over time
  - Allow customers to add their own configuration tools to extend what we offer…

freescale™

Processor Expert for QorIQ … For More Info

- Freescale's Processor Expert landing page
  - http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=PROCESSOR-EXPERT&tid=PEH
  - http://www.processorexpert.com/

- QorIQ Configuration Suite
  - http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=PE_QORIQ_SUITE&tid=PEH

- QorIQ Optimization Suite
  - http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=PE_QORIQ_OPTI_SUITE&tid=PEH

- Freescale Component Store – purchasing embedded software
  - http://www.freescale.com/webapp/sps/site/homepage.jsp?code=BEAN_STORE_MAIN&tid=SWnT


Processor Expert for QorIQ Configuration Suite Product Summary Page - Microsoft Internet Explorer provided by Freescale
Contact Us
Worldwide: United States
Login
My Freescale
Enter Part Number
Enter Keyword
Products
Applications
Design Resources
Support
Sample and Buy
About
Gregory's Freescale
Login
Annotate this Page
Browse History
My Recommendations
NEW
Page Actions
Freescale ▸ Processor Expert and Embedded Components ▸ Available Embedded Components ▸ PE_QORIQ_SUITE
Processor Expert for QorIQ Configuration Suite
Overview
Documentation
Downloads
Training & Support
The QorIQ Configuration Suite is designed to simplify the configuration of our most complex and powerful devices. The configuration suite is a set of tools for configuring the QorIQ devices to a known working state from silicon reset to avoid the typical board bring up issues that exist in most complex applications with custom hardware designs. This suite of tools is implemented as a set of components, each of which knows the details of the silicon control registers, configuration specifics, muxing rules and the necessary value ranges for all the configuration properties. Leveraging Processor Expert technology as a framework for these embedded components makes it easy to align tools with new silicon products and support upgraded models. Processor Expert technology, and therefore the Configuration Suite, is designed as a set of plug-ins that function within a standard Eclipse environment, such as CodeWarrior Development Studio.
Users will create a QorIQ Configuration Project that defines all imported and generated configuration files (and/or source code) so that a version control system can be used. A simple wizard is used to select basic configuration values and to define default settings. The Component Inspector window is used to modify each property. Each property is automatically evaluated to ensure that the values are correct and consistent with each other. Each tool has a set of criteria that is applied to ensure the configuration set is properly defined, including checks with the other tools for consistent configuration of the entire QorIQ silicon product. Each component generates output in the form necessary for configuration (PBL data in hex format, DDR configuration source code for u-boot, fmc xml data, Device Tree source files for dts, and so forth.). Every tool and property has tool tip style documentation aligned with the product manuals.
Supported Devices
P2040: QorIQ P2040/P2041 Low-End Quad-Core Communications Processors with Data Path
P3041: QorIQ P3041 Quad-Core Communications Processors with Data Path
P4040: QorIQ P4040 Quad-Core Communications Processors with Data Path
See All
Featured Documentation
QORIQCSPROCEXUG: Processor Expert Users Guide
QORIQCSGETSTARTEDUG: Configuration Suite Users Guide
QORIQCSINSTALLUG: Installation Users Guide
Current Updates and Releases
QorIQConfigSuite_3.5.x_Galileo: Configuration Suite Installer for Eclipse 3.5.x (Galileo)
QorIQConfigSuite_3.6.x_Helios: Configuration Suite Installer for Eclipse 3.6.x (Helios)
This Software Includes
Customer Application
Stacks
Applications
Drivers
APIs
Libraries
OS
BSP
HAL
Hypervisor
Processor
Config / Dev Tools
Eclipse Based Tools
Eclipse Compatibility
Eclipse-Galileo (v3.5.x)
Eclipse-Helios (v3.6.x)
CodeWarrior Development Studio
Training & Events
Processor Expert Orientation
Freescale Technology Forum
Connect With Us
Software Meets Silicon Blog
by Freescale's software experts
Feedback
See what other engineers are interested in:
What's This?

freescale™

Supporting the Most Sophisticated Customers…

FUTURE ELECTRONICS CO.LTD ADVA OPTICAL NETWORK AEONIUM SERVICES LTD ALCATEL-LUCENT ALCATEL-LUCENT AG ALVARION ARROW ELECTRONICS ASB AVNET ELECTRONICS BARCO N.V. CANOGA PERKINS CEAC INT'L CES CREATIVE ELECTRONIC CHANGWON UNIV. CISCO CISCO SYSTEMS CONTINOUS COMPUTING CRYPTO AG CYAN INC DIALOGIC, INC. DSPACE GMBH EDIXIA EMBEDDED SOLUTIONS EMCOSYS FUTURE ELECT INC GAMMA SP. Z O.O. GE INTELLEGENT PLATFORMS LTD GIGAMON SYSTEMS GUODIAN NANJING AUTOMATION HEIDENHAIN GMBH IEP GMBH INTERFACE CONCEPT IPWIRELESS ISKRATEL ELECTRONICS ITEC CONSULTING AS JUNIPER NETWORKS JUNIPER NETWORKS INDIA KDS KOLLMORGEN KONTRON MODULAR COMPUTERS KPIT CUMMINS INFOSYSTEMS LTD L-3 COMMUNICATIONS LOCKHEED MARTIN AERONAUTICS CO. MAGNETI MARELLI MOTORSPORT MERCURY COMPUTER SYSTEMS MIRANDA TECHNOLOGIES MOTOROLA MOBILITY INC. MYSTICAL ROSE TECHNOLOGIES NARI NETWORKS NKB VS PURESILICON, INC. REDCOM LABORATORIES INC ROHDE & SCHWARZ ROHDESCHWARZ GMBHCO. KG ROSS VIDEO SASET CHENGDU SERVERGY SERVERGY INC. SIMENA SOUTHERN FEDERAL UNIVERSITY SUSQUEHANNA INTERNATIONAL GROUP LLP TECHNOLOGY OF INFINITY THERMO FISHER SCIENTIFIC WB ELECTRONICS S.A. WINDRIVER WT MICROELECTRONICS CO. ZODIAC AEROSPACE

QorIQ Configuration Suite

Pricing & Availability

- Part numbers : CWA-QIQ-OPTP-FL (floating license) & CWA-QIQ-OPTP-NL (node locked)
- Price : $999 Annual Subscription
- License Duration : 1 year
- Support & Maintenance : Included
- Availability
  - Scenarios Tool – Now
  - DDRv – Now

freescale™

QorIQ Configuration Suite

Lab 1: Installing QCS



Installing QCS (should be done already)

- **D:\PEx Sessions\AMF-ENT-T1020 QCS\QCS233.zip**

- **Skip if you already have QCS v2.2 installed**
  - Setup for labs:
    - Create a directory where you have read/write permissions
      - Eg C:/QCS221 … from now on, we'll call this directory <qcs>
    - If you already have QCS installed, use that directory as <qcs>

- **Follow the instructions in**

  <qcs>/labs/ QORIQCSINSTALLUG.pdf

freescale™

Pre-boot Loader

Lab 2: Custom Hardware Configuration

Pre-boot Loader Hands-on

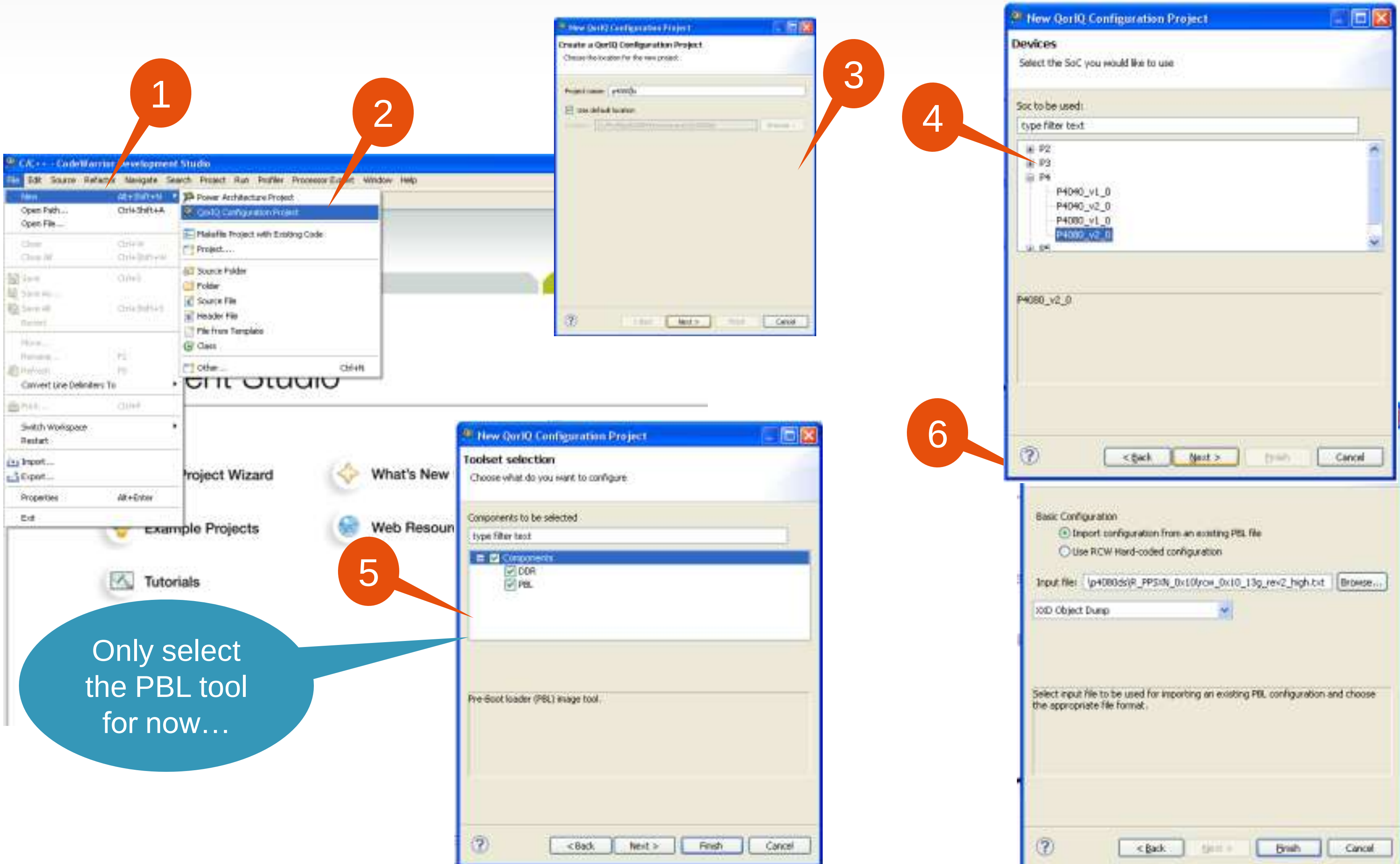
- **Step 1: Import and decode low speed config rcw_0x10_5g_rev2_low.bin.txt**
  - Platform clock : 600MHz
  - Core clock : 1.2GHz
  - FMAN1/2 clock : 450 MHz
  - DDR clock : 600MHz (1.2GHz)

- **Step 2: Use PBL tool to increase clock speed up to**
  - Platform clock : 800MHz
  - Core clock : 1.5GHz
  - FMAN1/2 clock : 600 MHz
  - DDR clock : 650MHz clock (1.3GHz)

- **Step 3: Use PBL tool to generate new RCW and compare outcomes with rcw_0x10_5g_rev2_high.bin.txt**

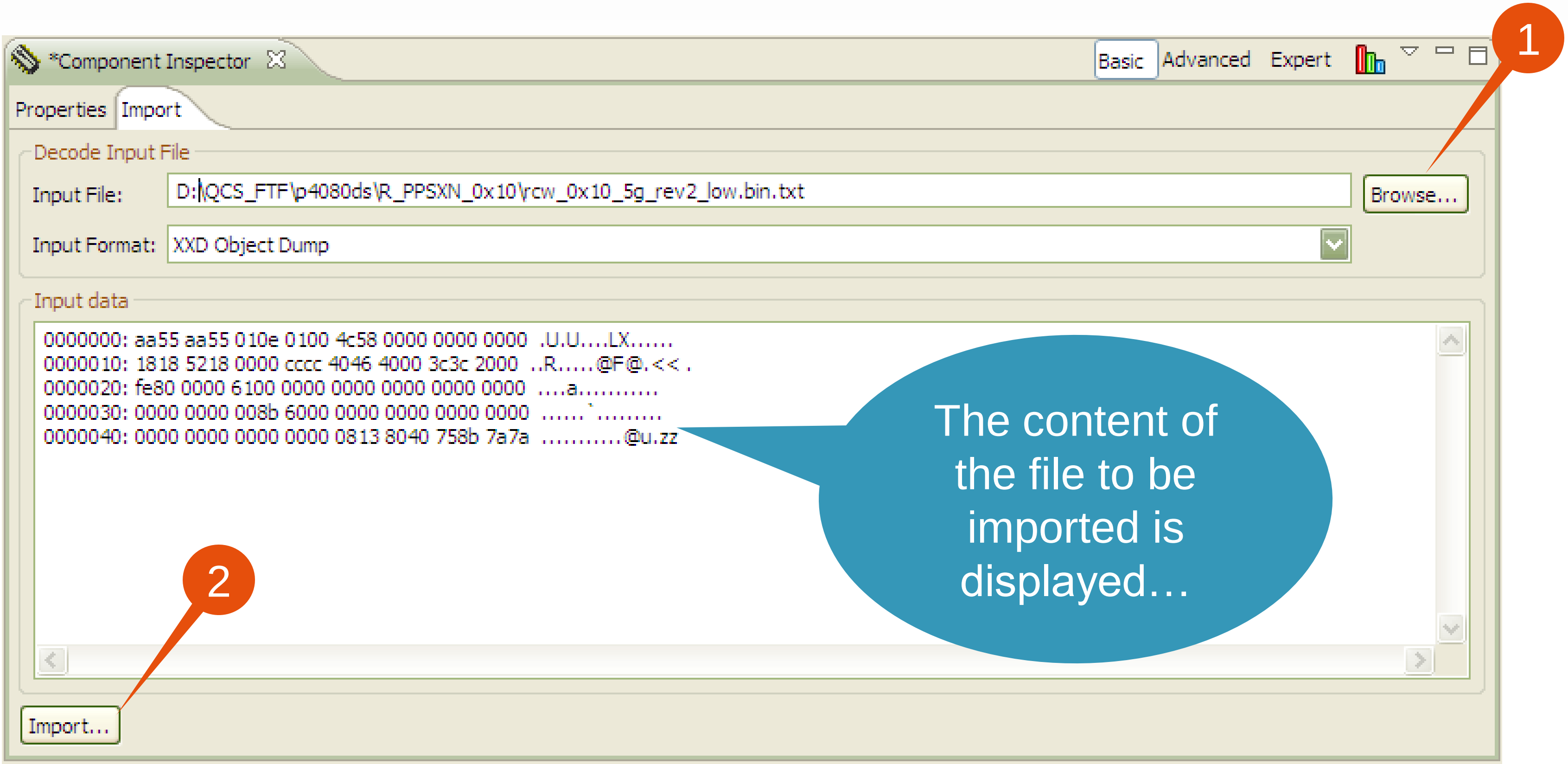
- **Step 4: Change Serdes Config to support 8 Gbe, compare result with rcw_0x16_all_rev2_high.bin**

Pre-Boot Loader Step 1: Create a New Project

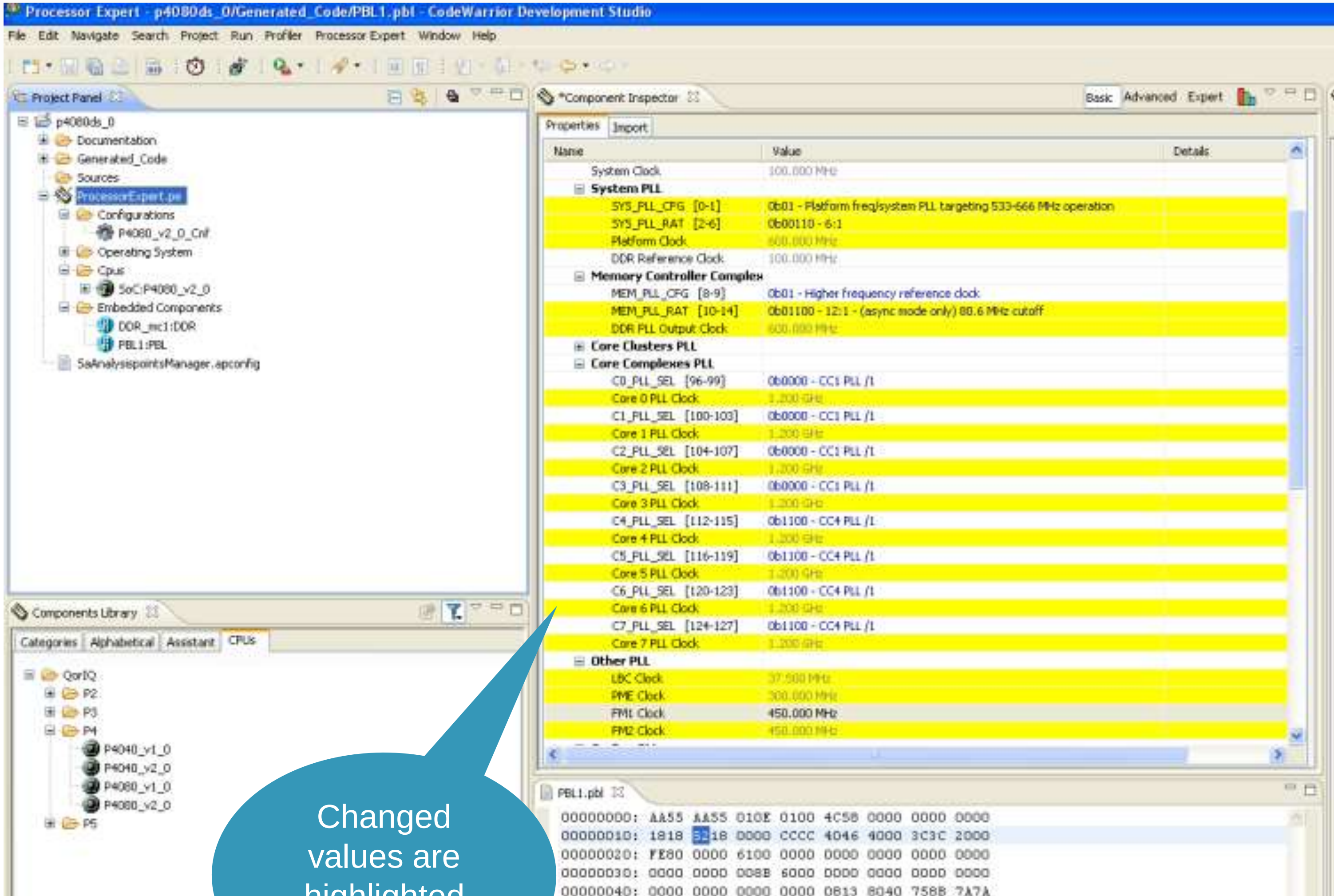


freescale™

Pre-boot Loader Step 2a:
Import rcw_0x10_5g_rev2_low.bin.txt



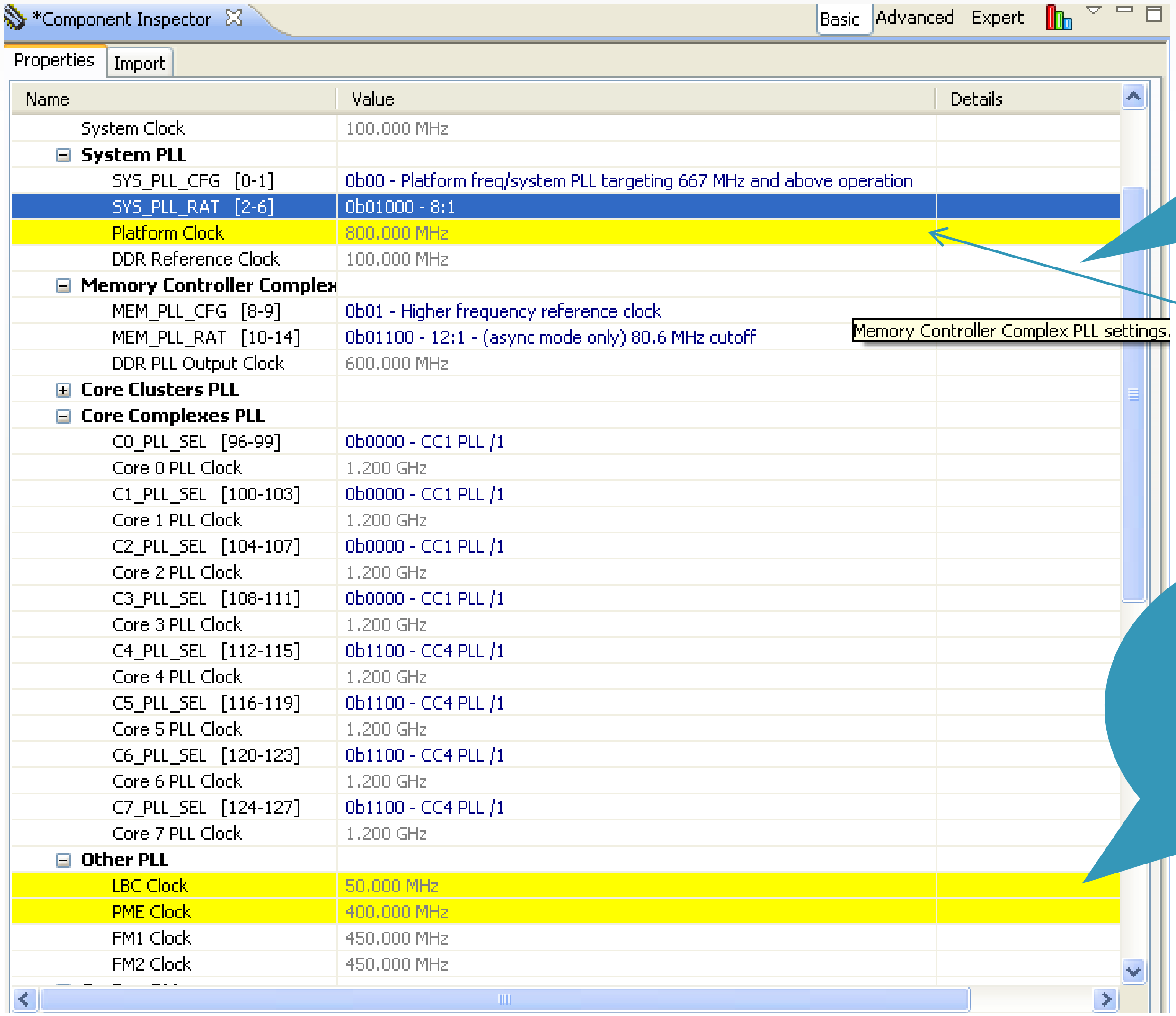
Pre-boot Loader Step 2b: See Differences Highlighted



Changed values are highlighted after import.

freescale™

Pre-boot Loader Step 3: Increase Platform Clock



Name	Value	Details
System Clock	100.000 MHz	
⊟ **System PLL**		
SYS_PLL_CFG [0-1]	0b00 - Platform freq/system PLL targeting 667 MHz and above operation	
SYS_PLL_RAT [2-6]	0b01000 - 8:1	
Platform Clock	800.000 MHz	
DDR Reference Clock	100.000 MHz	
⊟ **Memory Controller Complex**		
MEM_PLL_CFG [8-9]	0b01 - Higher frequency reference clock	
MEM_PLL_RAT [10-14]	0b01100 - 12:1 - (async mode only) 80.6 MHz cutoff	
DDR PLL Output Clock	600.000 MHz	
⊞ **Core Clusters PLL**		
⊟ **Core Complexes PLL**		
C0_PLL_SEL [96-99]	0b0000 - CC1 PLL /1	
Core 0 PLL Clock	1.200 GHz	
C1_PLL_SEL [100-103]	0b0000 - CC1 PLL /1	
Core 1 PLL Clock	1.200 GHz	
C2_PLL_SEL [104-107]	0b0000 - CC1 PLL /1	
Core 2 PLL Clock	1.200 GHz	
C3_PLL_SEL [108-111]	0b0000 - CC1 PLL /1	
Core 3 PLL Clock	1.200 GHz	
C4_PLL_SEL [112-115]	0b1100 - CC4 PLL /1	
Core 4 PLL Clock	1.200 GHz	
C5_PLL_SEL [116-119]	0b1100 - CC4 PLL /1	
Core 5 PLL Clock	1.200 GHz	
C6_PLL_SEL [120-123]	0b1100 - CC4 PLL /1	
Core 6 PLL Clock	1.200 GHz	
C7_PLL_SEL [124-127]	0b1100 - CC4 PLL /1	
Core 7 PLL Clock	1.200 GHz	
⊟ **Other PLL**		
LBC Clock	50.000 MHz	
PME Clock	400.000 MHz	
FM1 Clock	450.000 MHz	
FM2 Clock	450.000 MHz	

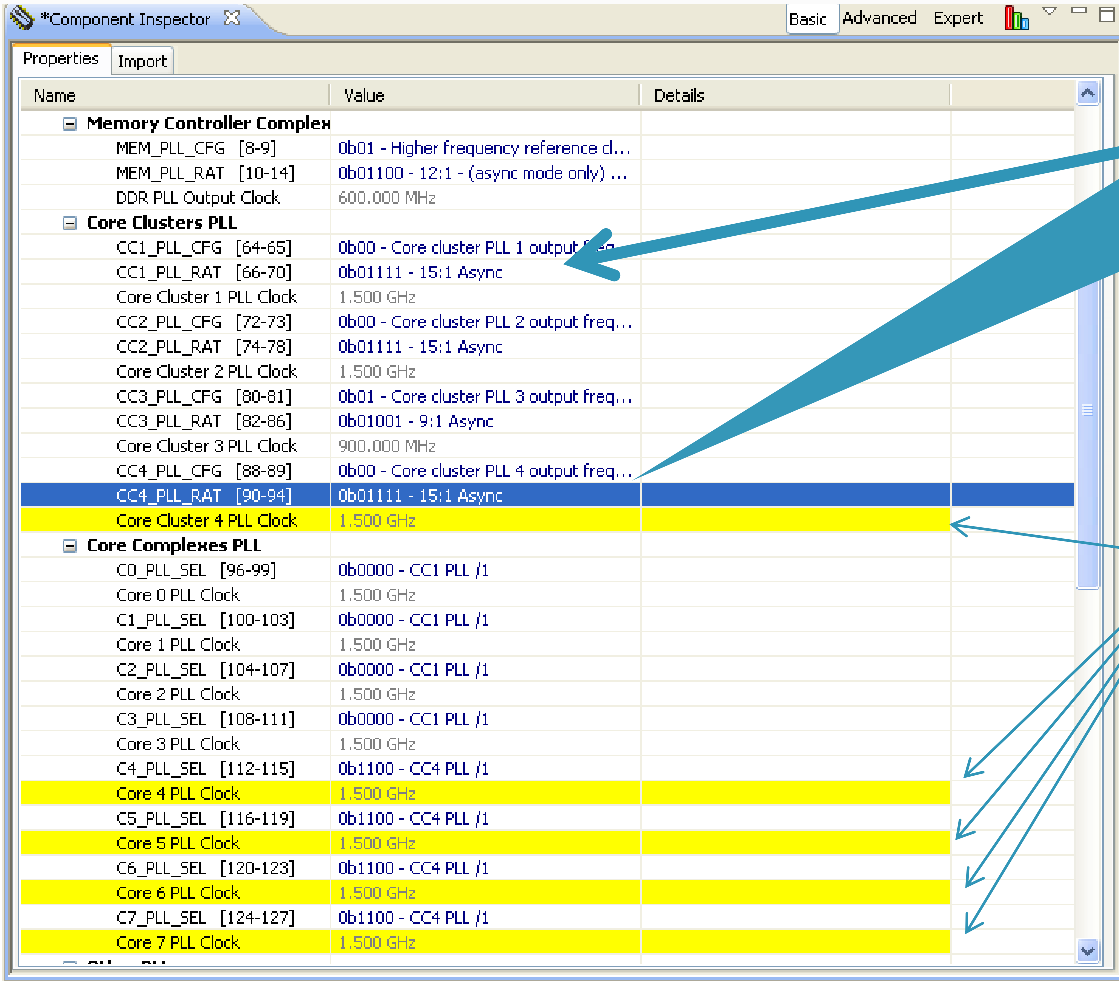
Changing the ratio to 8:1 makes platform clock 800 MHz

600 Mhz → 800 Mhz

And also changes the LBC and PME clocks

freescale™

Pre-boot Loader: Change Clock Ratios



Changing the CC1/CC4 PLL_RAT for cores to 15:1 sets core clocks to 1.5 Ghz

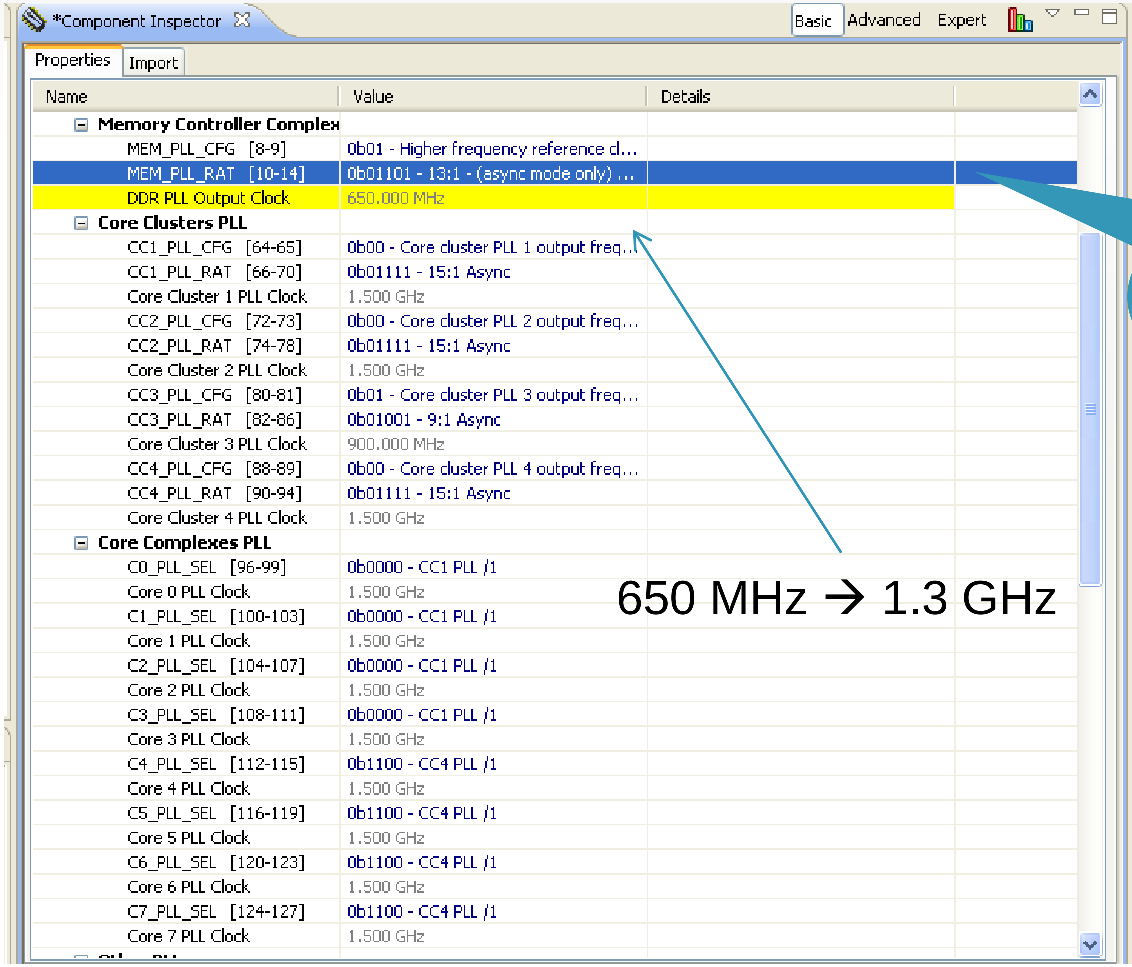
1.2 Ghz → 1.5 Ghz

CC1 clocks core 0-3
CC4 clocks core 4-7

Note: Please change CC2 also, even if unused, to reach exact high rcw

freescale™

Pre-boot Loader: Increase DDR Output Clock



650 MHz → 1.3 GHz

Set MEM_PLL_RAT to 13:1

freescale™

Pre-boot Loader Step 6: Increase FMAN1/2 Speed

*Component Inspector

Properties | Import

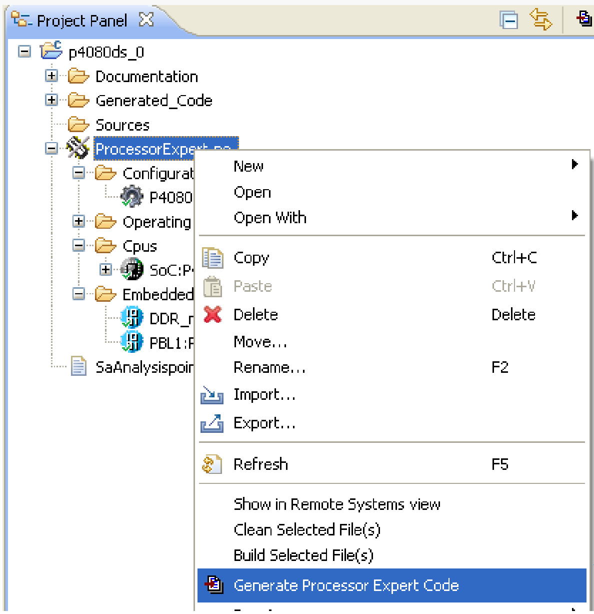
Name	Value	Details
Memory Controller Complex		
MEM_PLL_CFG [8-9]	0b01 - Higher frequency reference cl...	
MEM_PLL_RAT [10-14]	0b01101 - 13:1 - (async mode only) ...	
DDR PLL Output Clock	650.000 MHz	
Core Clusters PLL		
CC1_PLL_CFG [64-65]	0b00 - Core cluster PLL 1 output freq...	
CC1_PLL_RAT [66-70]	0b01111 - 15:1 Async	
Core Cluster 1 PLL Clock	1.500 GHz	
CC2_PLL_CFG [72-73]	0b00 - Core cluster PLL 2 output freq...	
CC2_PLL_RAT [74-78]	0b01111 - 15:1 Async	
Core Cluster 2 PLL Clock	1.500 GHz	
CC3_PLL_CFG [80-81]	0b00 - Core cluster PLL 3 output freq...	
CC3_PLL_RAT [82-86]	0b01100 - 12:1 Async	
Core Cluster 3 PLL Clock	1.200 GHz	
CC4_PLL_CFG [88-89]	0b00 - Core cluster PLL 4 output freq...	
CC4_PLL_RAT [90-94]	0b01111 - 15:1 Async	
Core Cluster 4 PLL Clock	1.500 GHz	
Core Complexes PLL		
Other PLL		
LBC Clock	50.000 MHz	
PME Clock	400.000 MHz	
FM1 Clock	600.000 MHz	
FM2 Clock	600.000 MHz	
SerDes PLL		
SerDes PLL and Protocol Config		
Misc. PLL-Related Configuration		
DDR_SYNC [184]	0b0 - Both DDRs in asynchronous mode	
Boot Configuration		
Clocking Configuration		
PME_CLK_SEL [224]	0b0 - Platform Clock /2	
FM1_CLK_SEL [225]	0b1 - Core Cluster PLL 3 /2	
FM2_CLK_SEL [226]	0b1 - Core Cluster PLL 3 /2	
DRAM_LAT [230-231]	0b01 - 8-8-8, 9-9-9, 10-10-10, 11-11...	
DDR_RATE [232]	0b0 - Refer to hardware specification...	

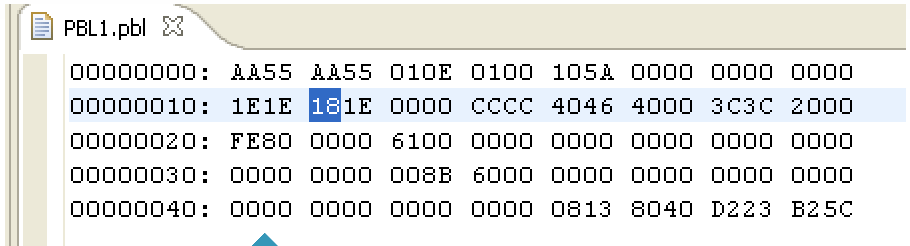
Set CC3_PLL_RAT to 12:1 to set Frame Manager clocks

450 MHz → 600 MHz

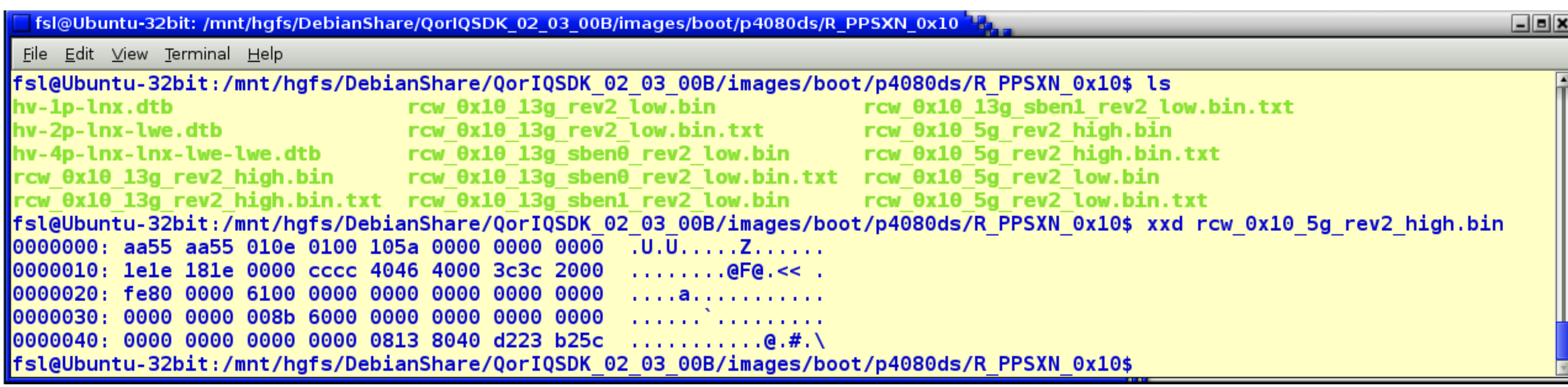
freescale™

Pre-boot Loader Step 7 – Generate Upgraded RCW



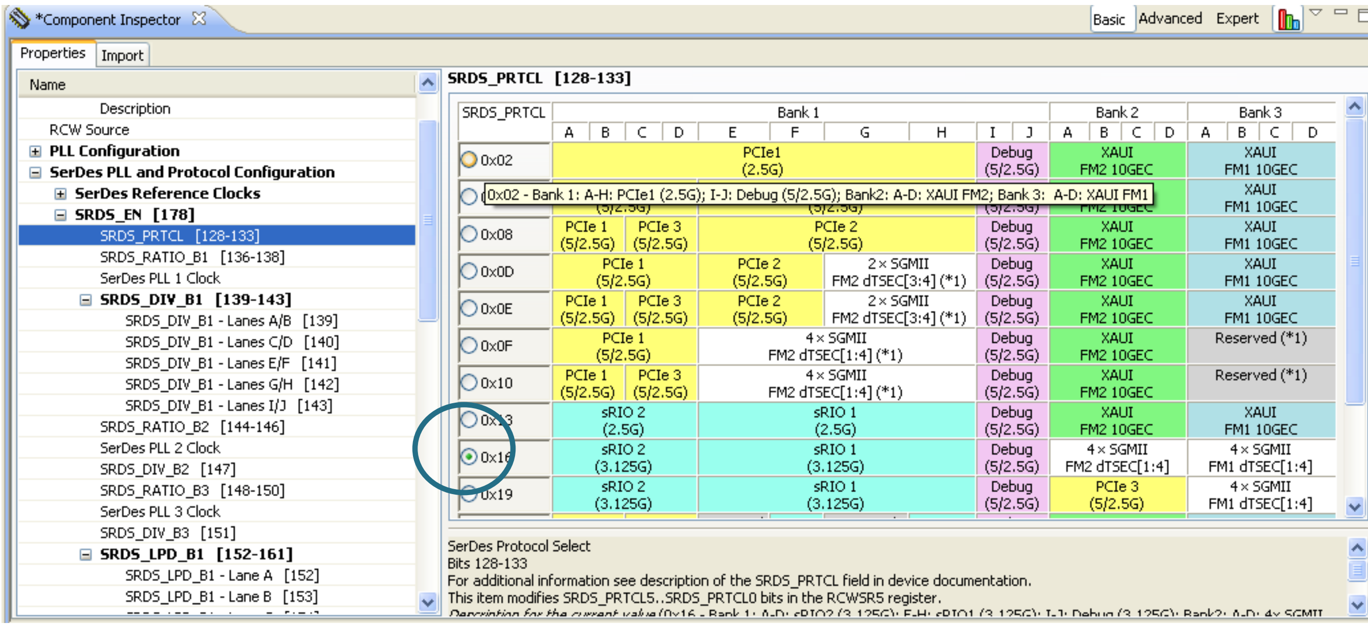


Generating the PBL provides the same RCW as the high-speed RCW seen in the SDK



Pre-boot Loader Step 8: Change Serdes for 2 x 4 SGMII

- Starting from previous rcw_0x10_5g_rev2_high.bin.txt config, let's adapt Serdes protocol configuration to allow 2x4 SGMII



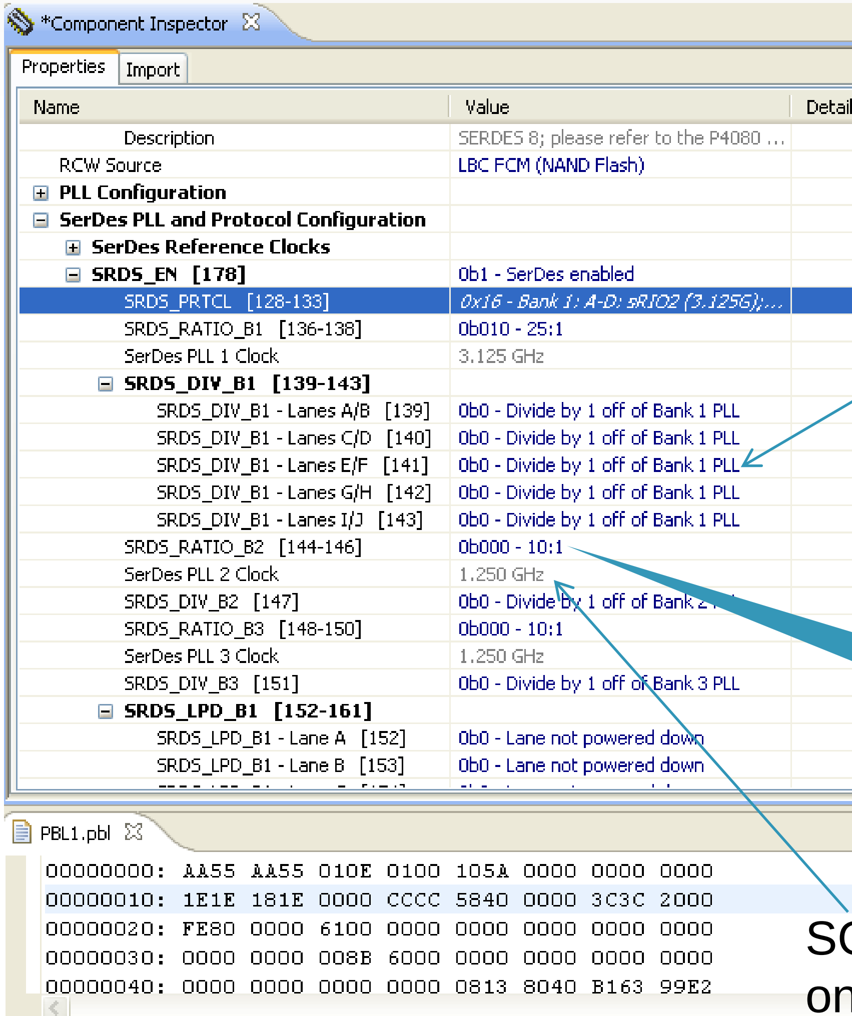
SRDS_PRTCL 0x10 to 0x16

Table 5. SerDes Lane Multiplexing/Configuration

Bank 1										Bank 2				Bank 3			
A	B	C	D	E	F	G	H	I	J	A	B	C	D	A	B	C	D
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	14	16	17
SLOT 1		SLOT 2		SLOT 3				Aurora Conn.		SLOT 4				SLOT 5			

P4080DS slots as per user manual

Pre-boot Loader Step 9: Adapt Serdes Clocks



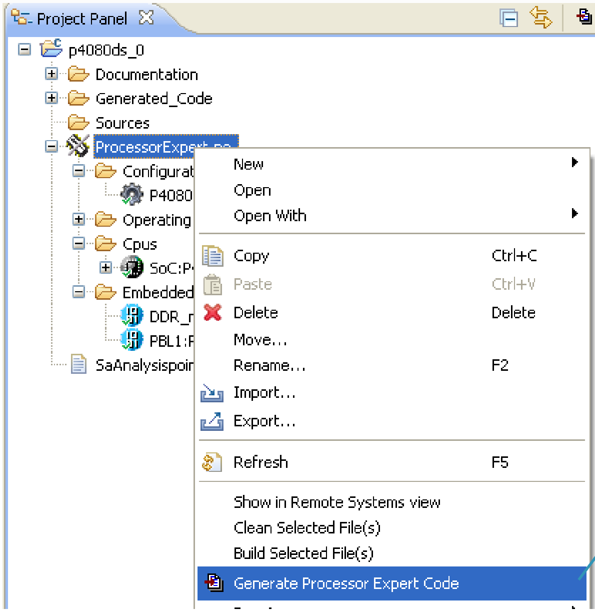
SRIO requires **3.125 Ghz** on Bank1 lane E,F,G,H.
Set to Divide by 1.

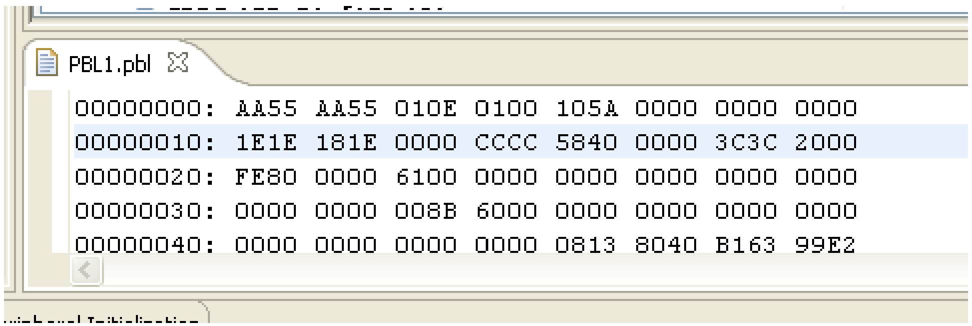
Set SRDS_RATIO_B2 to 10:1

SGMII requires **1.250 GHz** on Bank2 and Bank3

freescale™

PBL Step 9: Generate Pre-boot Loader and Compare with the RCW Provided in the SDK





Generating the PBL now provides us the same RCW than the RCW seen in the SDK2.3

```
fsl@Ubuntu-32bit:/mnt/hgfs/DebianShare/QorIQSDK_02_03_00B/images/boot/p4080ds/R_RRRSS_0x16$ ls
hv-1p-lnx-agent.dtb  hv-1p-lnx-host.dtb          rcw_0x16_all_rev2_high.bin      rcw_0x16_all_rev2_low.bin
hv-1p-lnx.dtb        hv-4p-lnx-lnx-lwe-lwe.dtb  rcw_0x16_all_rev2_high.bin.txt  rcw_0x16_all_rev2_low.bin.txt
fsl@Ubuntu-32bit:/mnt/hgfs/DebianShare/QorIQSDK_02_03_00B/images/boot/p4080ds/R_RRRSS_0x16$ xxd rcw_0x16_all_rev2_high.bin
0000000: aa55 aa55 010e 0100 105a 0000 0000 0000  .U.U.....Z......
0000010: 1e1e 181e 0000 cccc 5840 0000 3c3c 2000  ........X@..<< .
0000020: fe80 0000 6100 0000 0000 0000 0000 0000  ....a...........
0000030: 0000 0000 008b 6000 0000 0000 0000 0000  ......`.........
0000040: 0000 0000 0000 0000 0813 8040 b163 99e2  ...........@.c..
fsl@Ubuntu-32bit:/mnt/hgfs/DebianShare/QorIQSDK_02_03_00B/images/boot/p4080ds/R_RRRSS_0x16$
```

freescale™

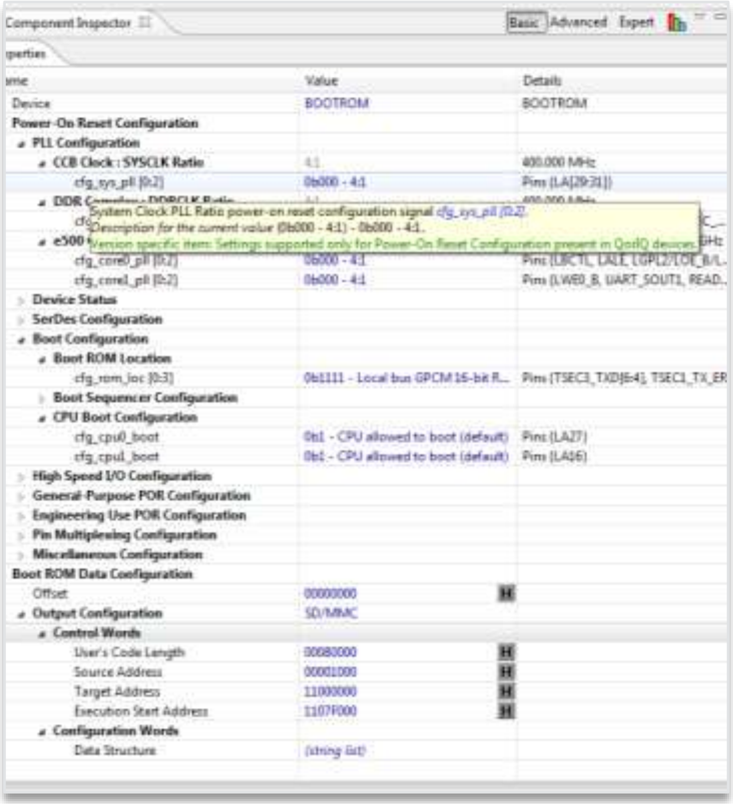
BOOTROM Configuration

Lab 3: Custom Hardware Configuration

BOOTROM Configuration tool

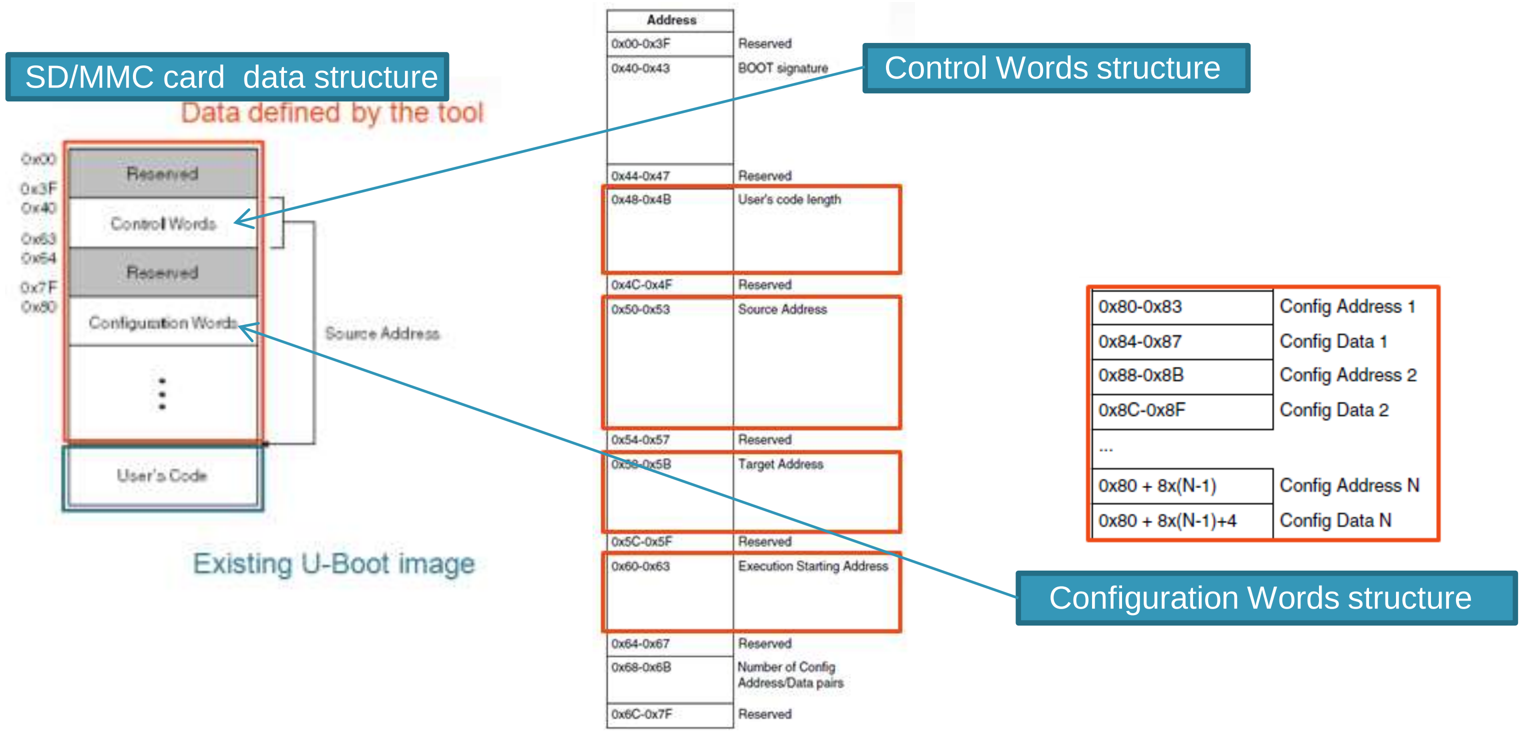
Overview

- Helps in Power-on Reset (POR) device configuration by definition of values for POR configuration signals and generates overview report including POR value required for each POR configuration pin (device specific)
- Helps in building a configuration file to be used in a boot image creation process (*boot_format*) for various memory interfaces



BOOTROM Configuration tool

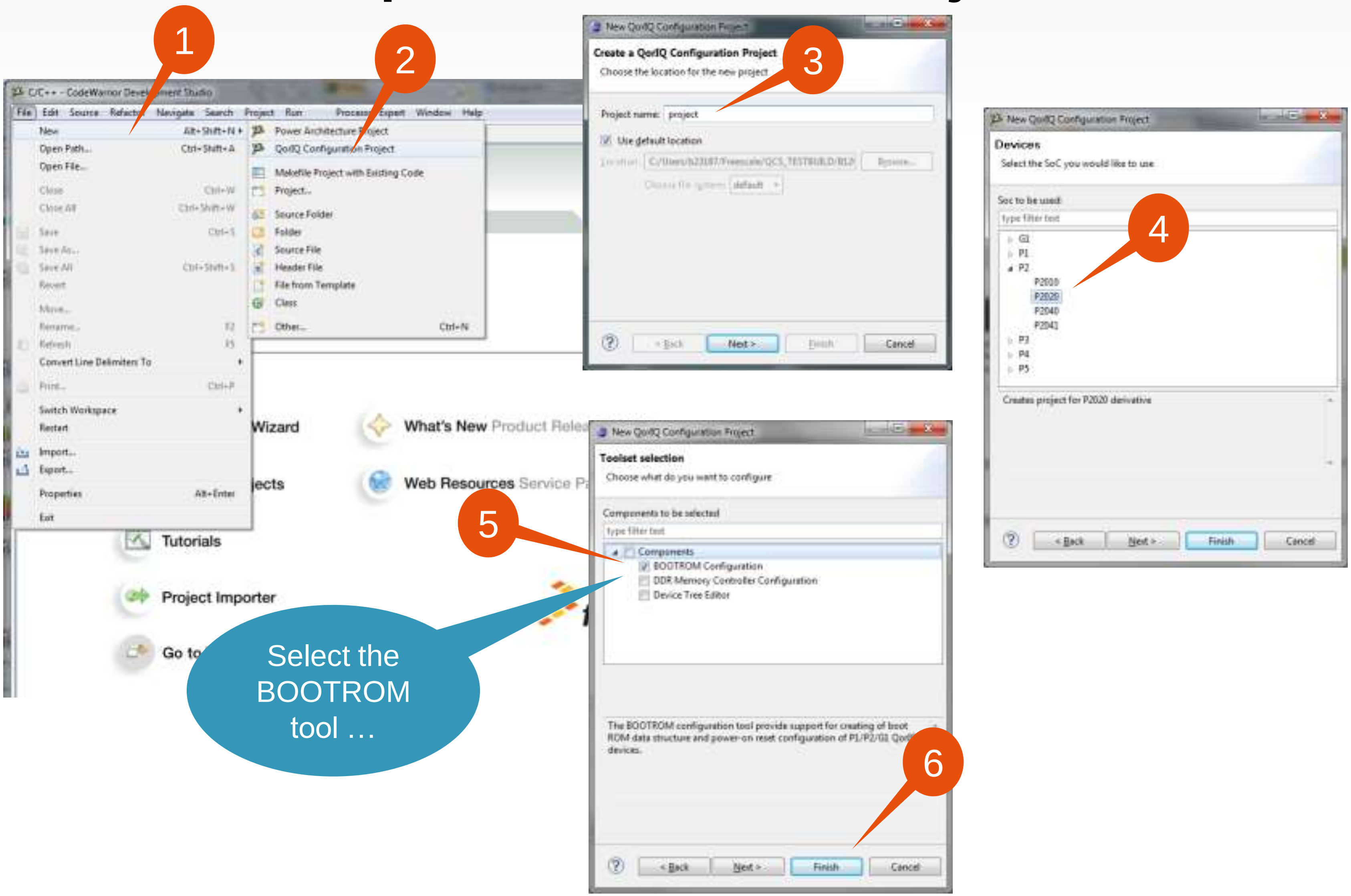
- *Power-on Reset (POR) configuration signal value* – binary value 0b0 represents a signal pulled down to GND and a value 0b1 represents a signal pulled up to Vdd, regardless of the sense of the functional signal name on the signal
- *Configuration file* – data structure including control and configuration words, two parts that needs to be put together with user's code (typically u-boot image) to create booting image for a device



BOOTROM Hands-on

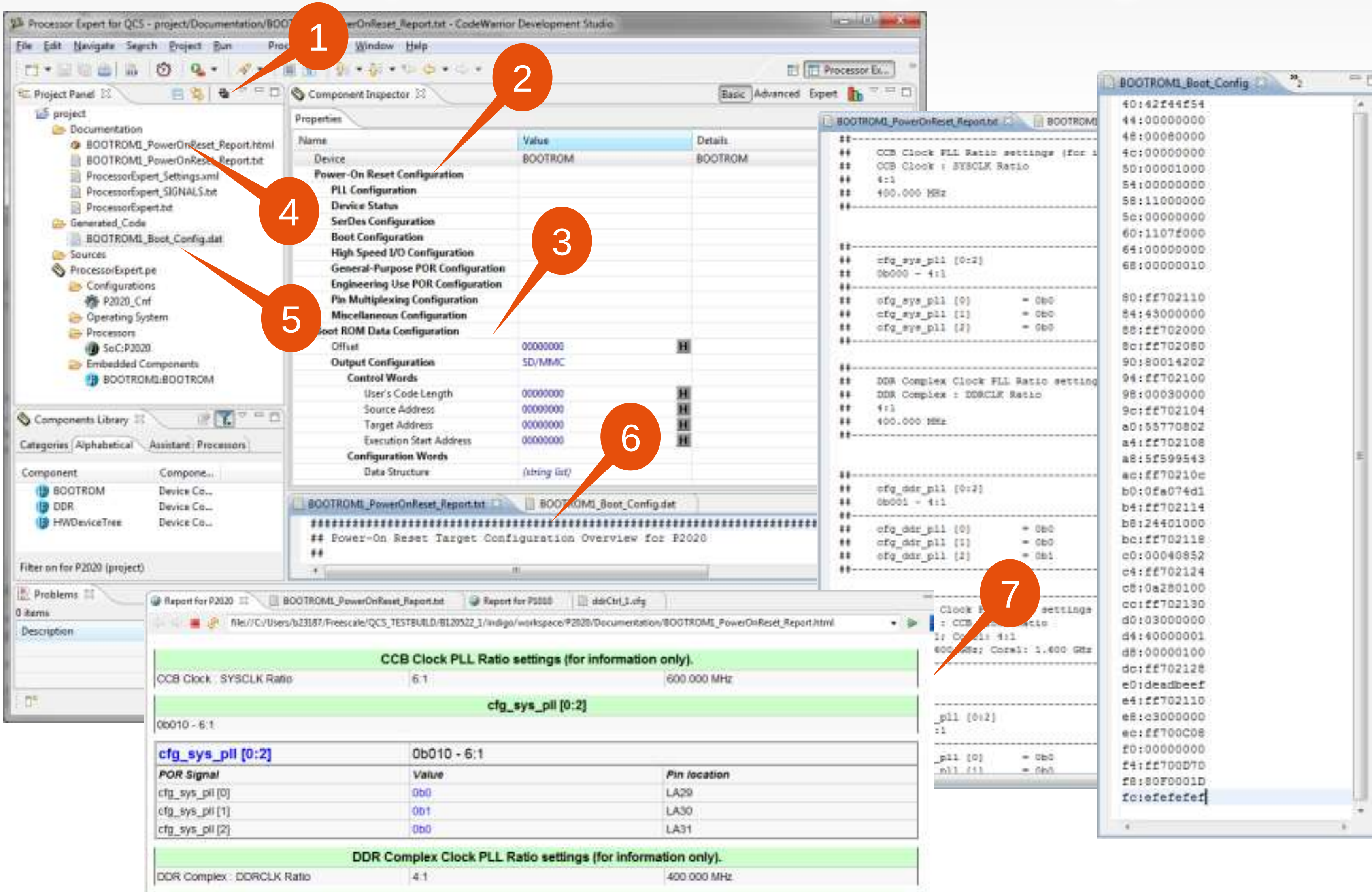
- Step 1: Create configuration project for the P2020 device
- Step 2: Use BOOTROM tool to review & change settings
  - Details:
    - Change CCB and core clocking – 600 / 900 MHz
    - I/O port configuration – required interfaces: PCIe, SRIO and eTSEC
    - boot location – set to boot from SD/MMC card
- Step 3: Observe Power-on Reset overview details
    - Generated overview report – txt/HTML
- Step 4: Use BOOTROM tool to prepare configuration data file for boot image processing
    - Generated configuration file (.dat) – support of booting from on-chip ROM (e.g. eSDHC or eSPI), base data file for boot image processing using external booting utility application
- Step 5: Usage of configuration data file with external booting utility application (*using boot_format - this application is a part of BSP release*)

BOOTROM Step 1: Create a New Project

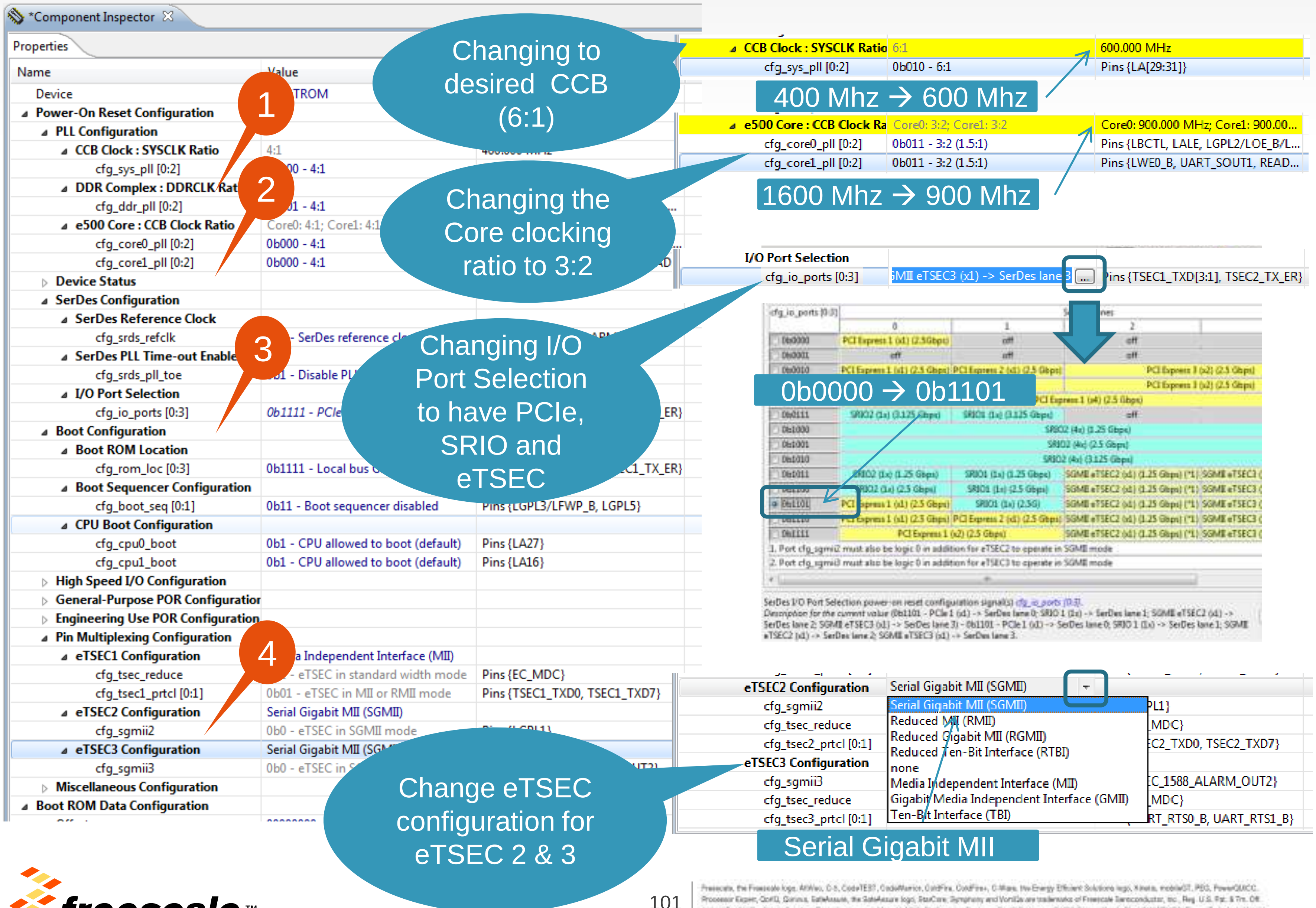


freescale™

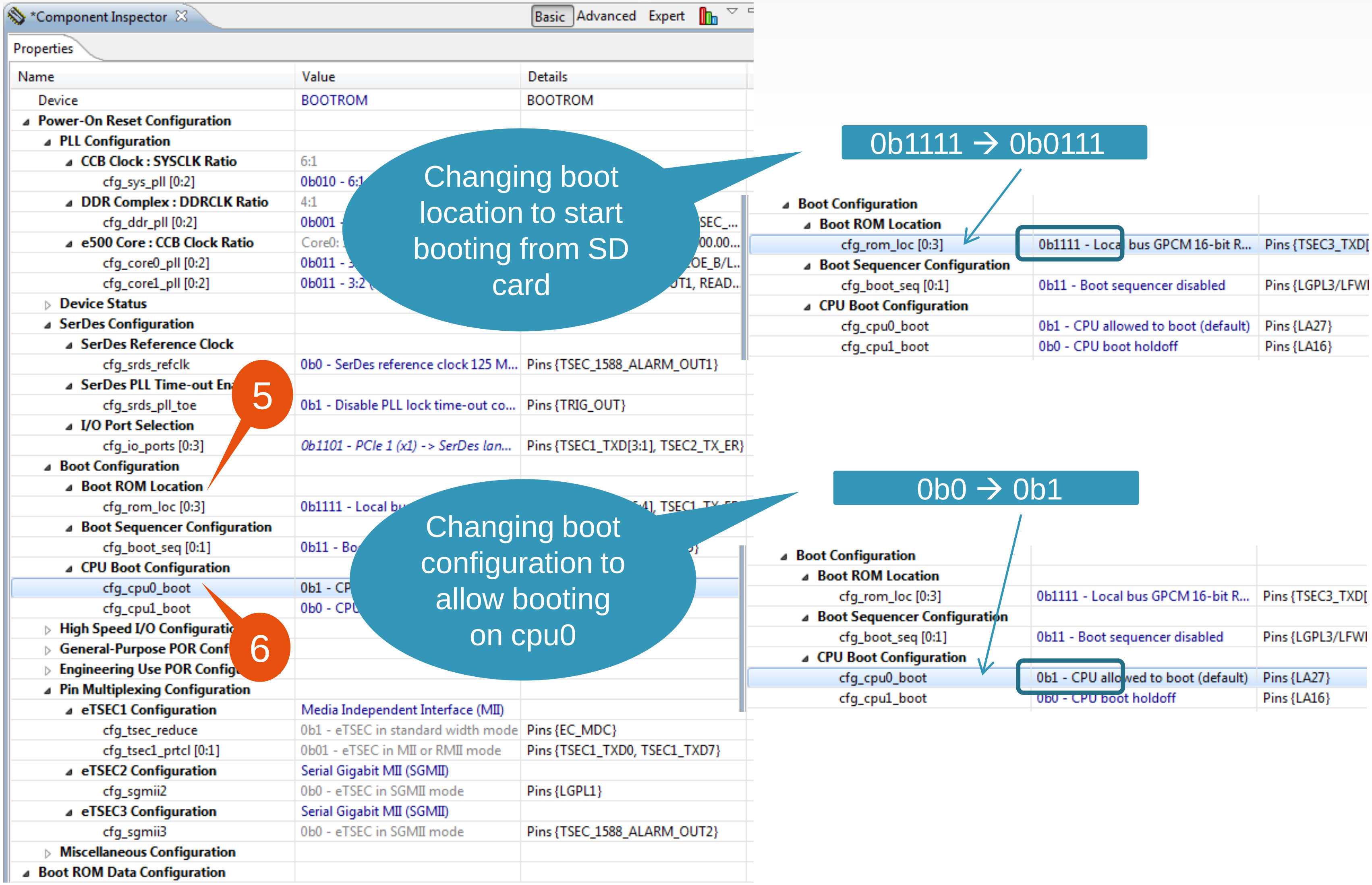
BOOTROM Step 2: Review & change settings …



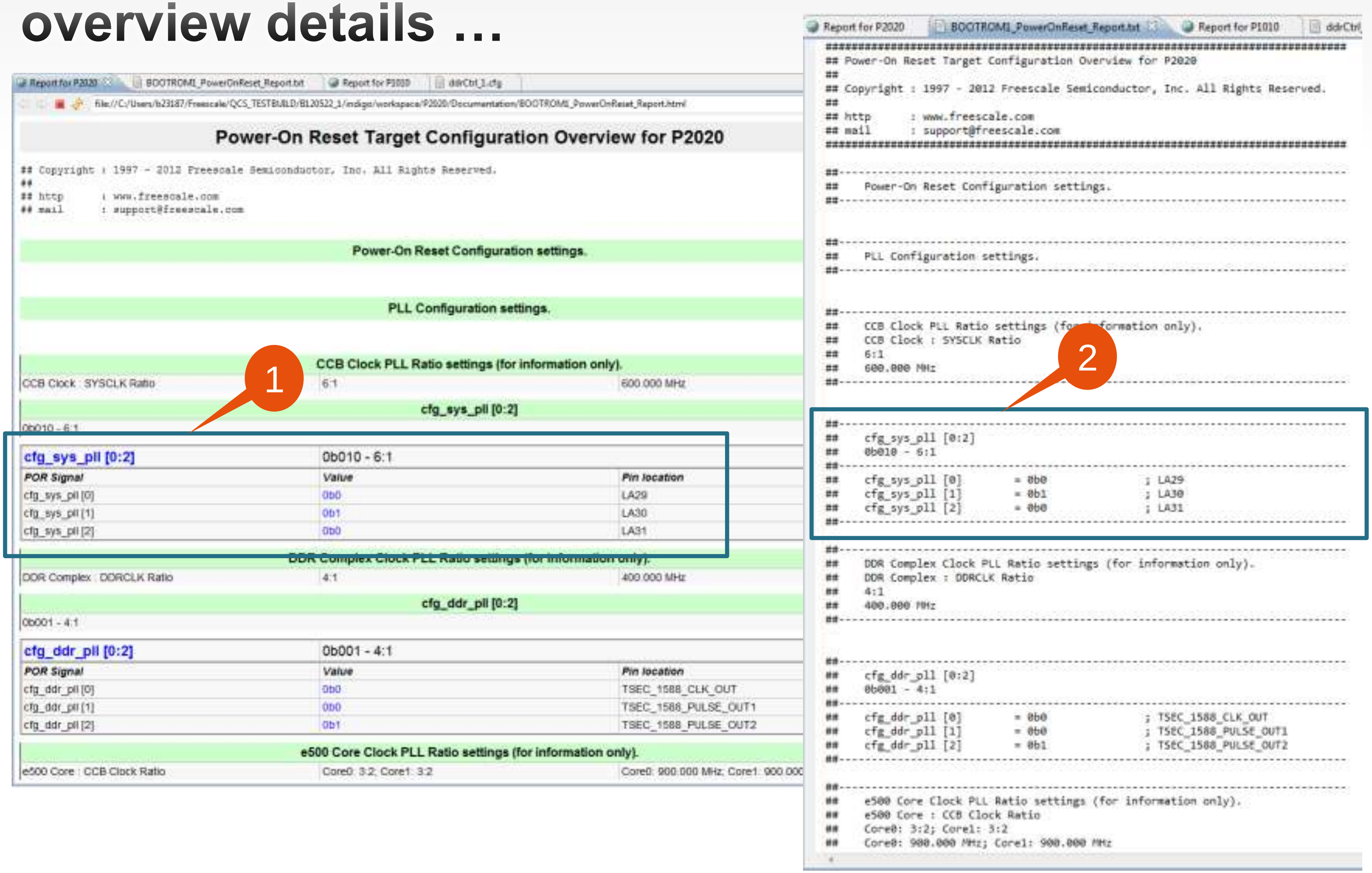
BOOTROM Step 2: Review & change settings …



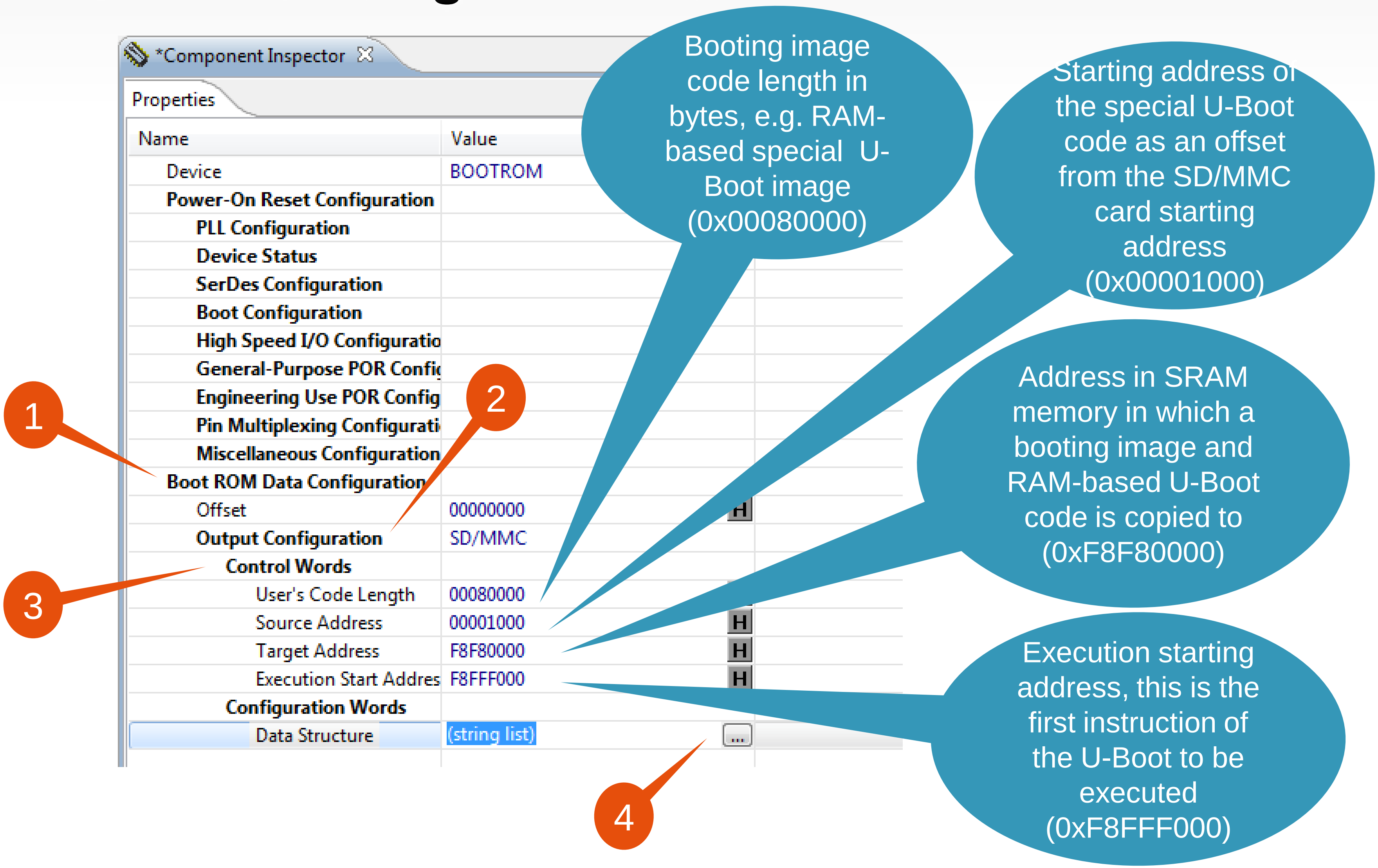
BOOTROM Step 2: Configuration for SD booting



BOOTROM Step 3: Observe Power-on reset overview details …

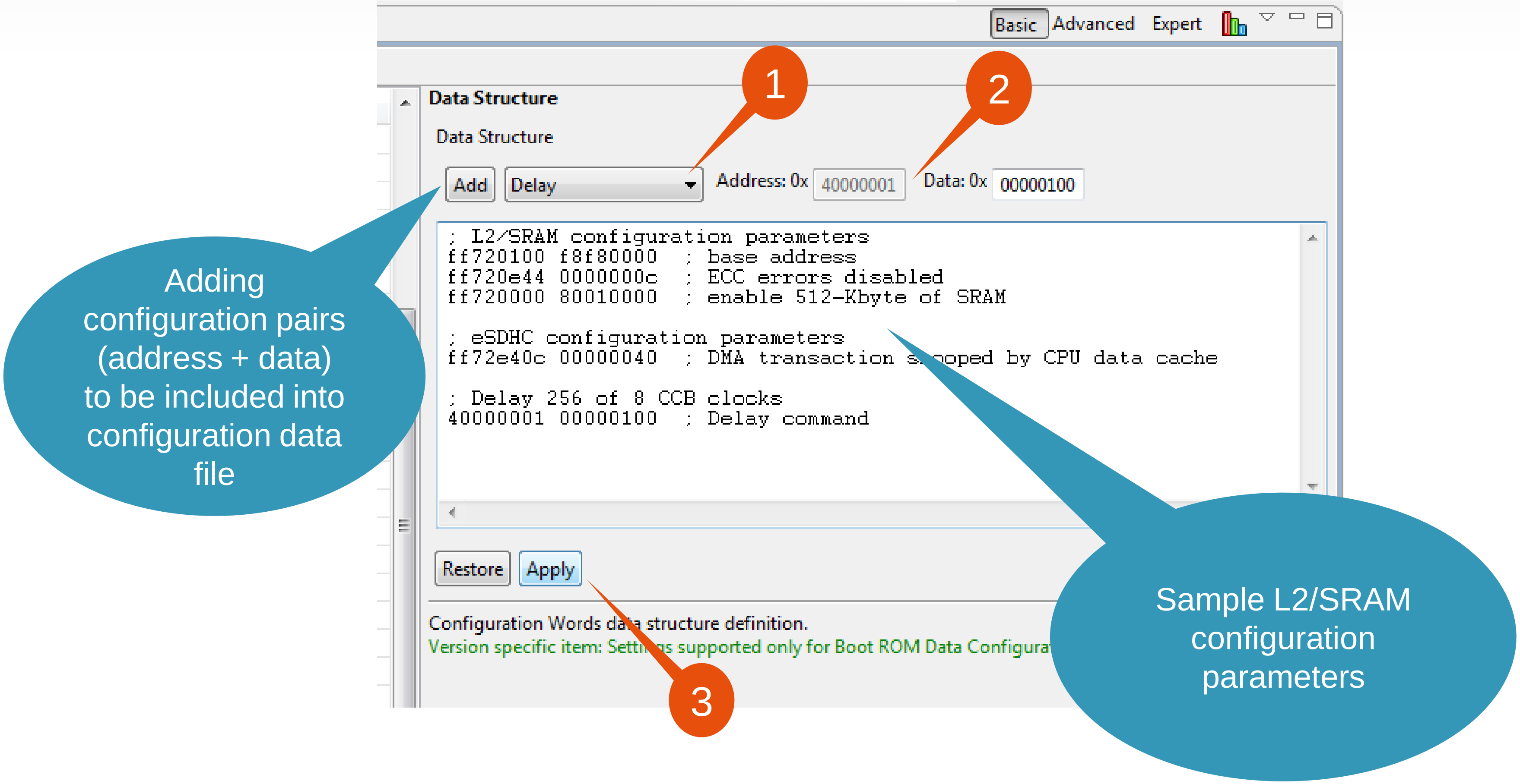


BOOTROM Step 4: Prepare configuration data file for SD/MMC using L2 cache



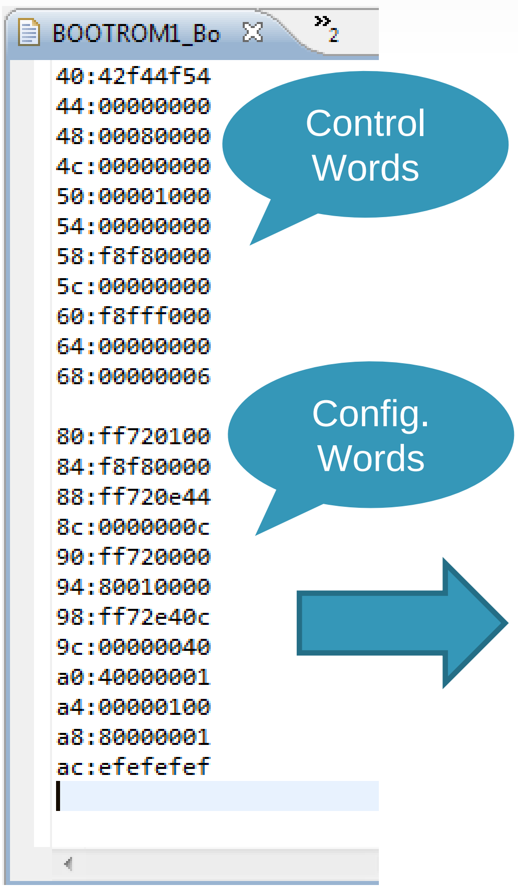
freescale™

BOOTROM Step 4: Prepare configuration data file for SD/MMC using L2 cache



freescale™

BOOTROM Step 5: Usage of configuration data file with external booting utility application



Use a booting utility *boot_format* from a BSP package for P2020RDB/P102RDB

Once you have installed BSP and let configured ltib to build root file system *rootfs.tar.gz.uboot* for Linux boot. Boot the board using this root file system and *boot_format* utility can be located under:

[root@P1020RDB /]# *cd /boot_format/*
[root@P1020RDB */boot_format]# ls -l*

```
-rwxr-xr-x 1 root root 10400 Apr 7 2010 boot_format
-rw-r--r-- 1 root root   530 Apr 7 2010 config_sram.dat
```

The utility shows how to use it when typing ***./boot_format***

```
Usage: ./boot_format config_file image -sd dev [-o out_config] |
-spi  spiimage

config_file : includes boot signature and config words
image       : the U-Boot image for booting from eSDHC/eSPI
dev         : SDCard's device node(e.g. /dev/sdb, /dev/mmcblk0)
spiimage    : boot image for SPI mode
out_config  : modified config file for SD mode
```

BOOTROM tool generated config file can be used together with U-Boot image and put on an SD/MMC card using command line, e.g. for /dev/mmcblk0:

```
./boot_format BOOTROM1_Boot_Config.dat u-boot.bin -sd /dev/mmcblk0
```

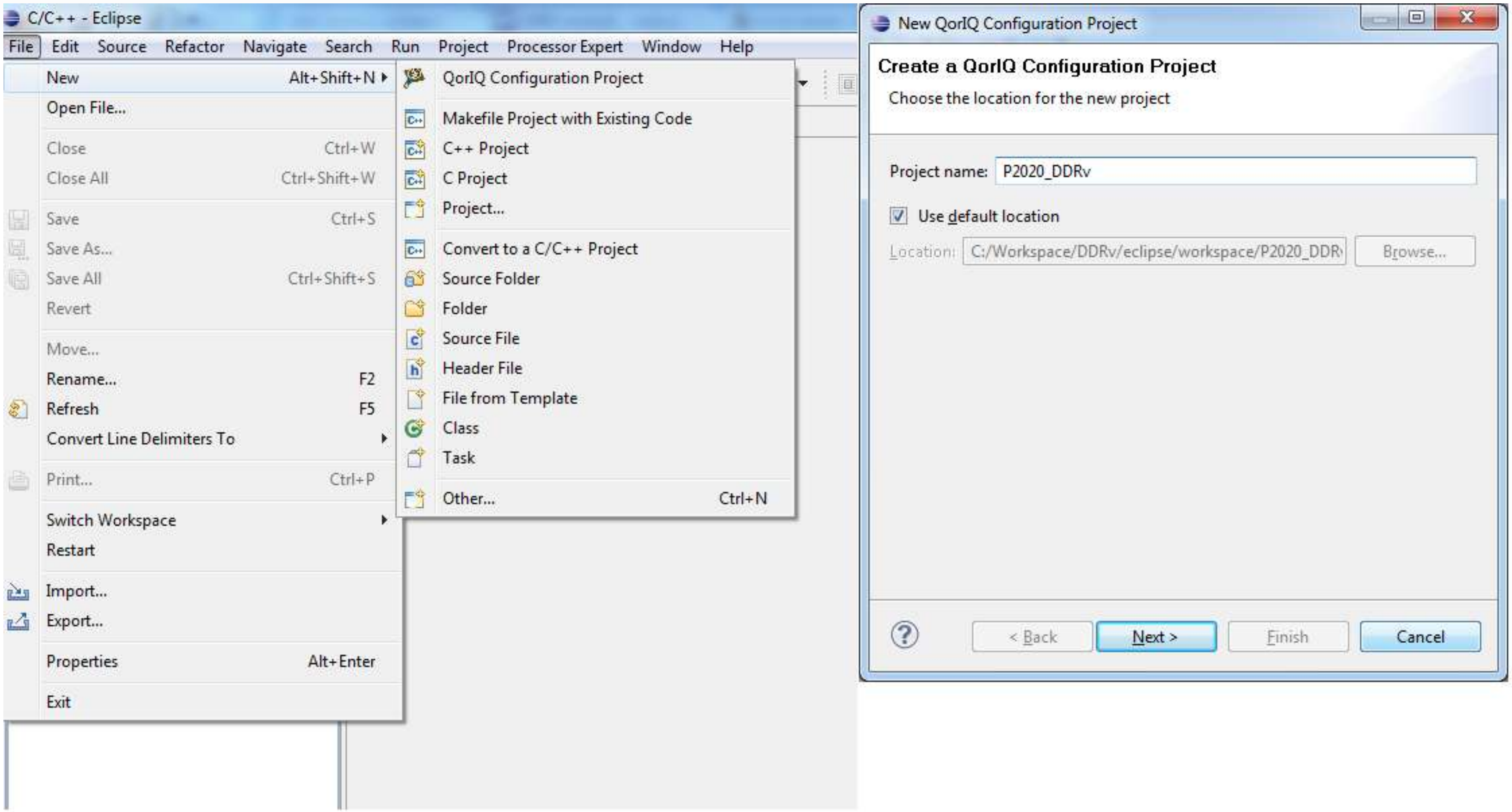
freescale™

DDR Configuration

Lab 4a: Setup DDR for P2020

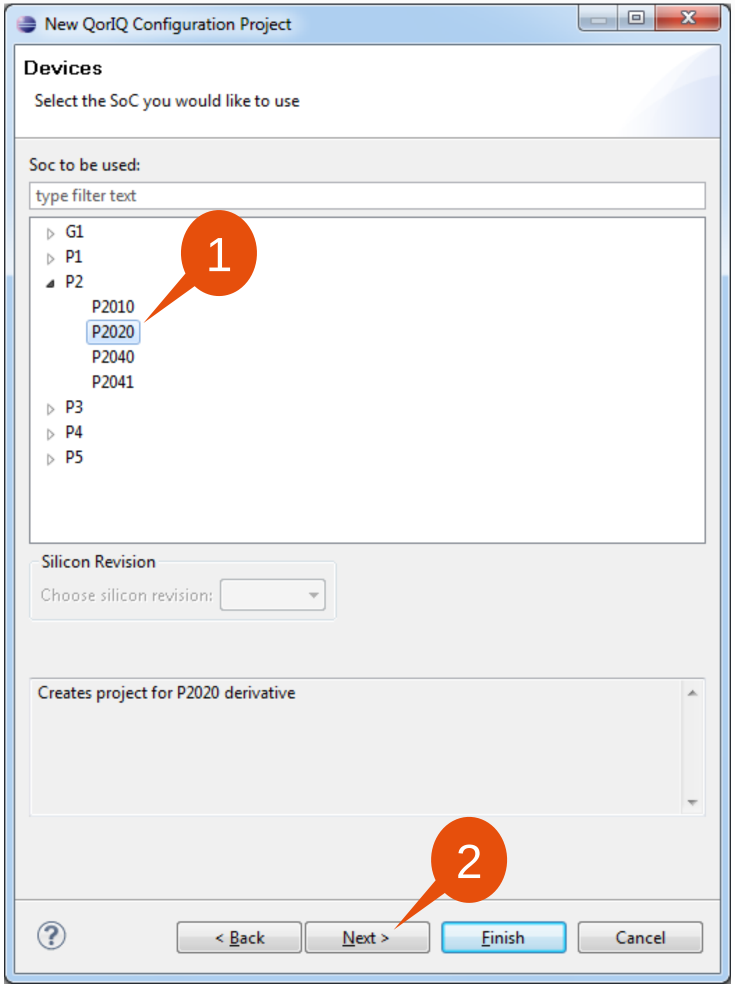


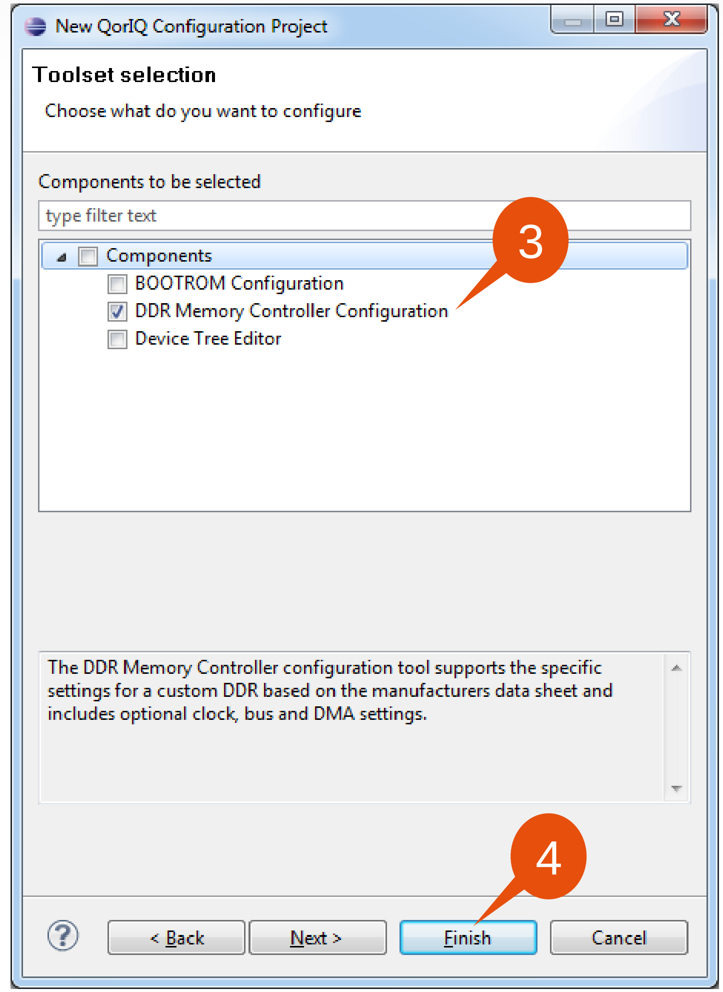
Create New QCS project



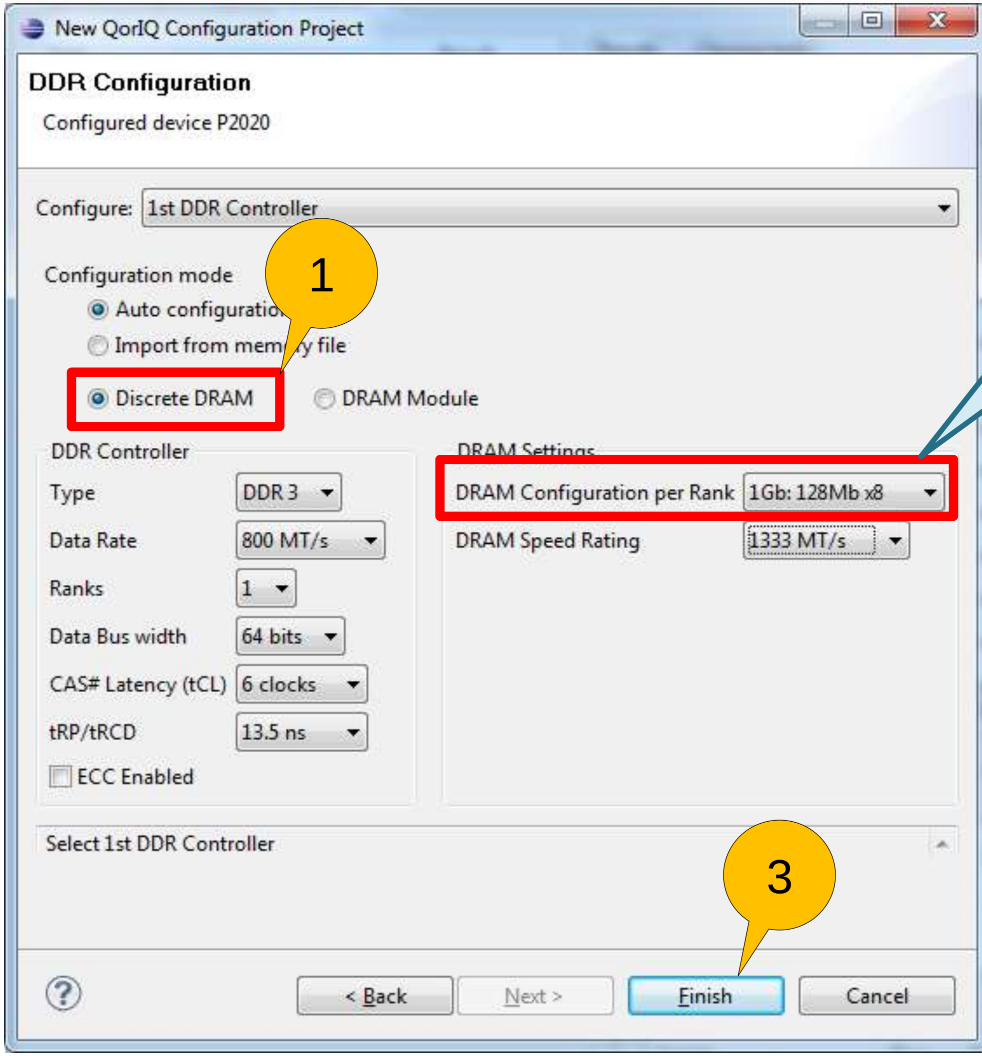
freescale™

Select device and DDR component





DDR New Project Wizard

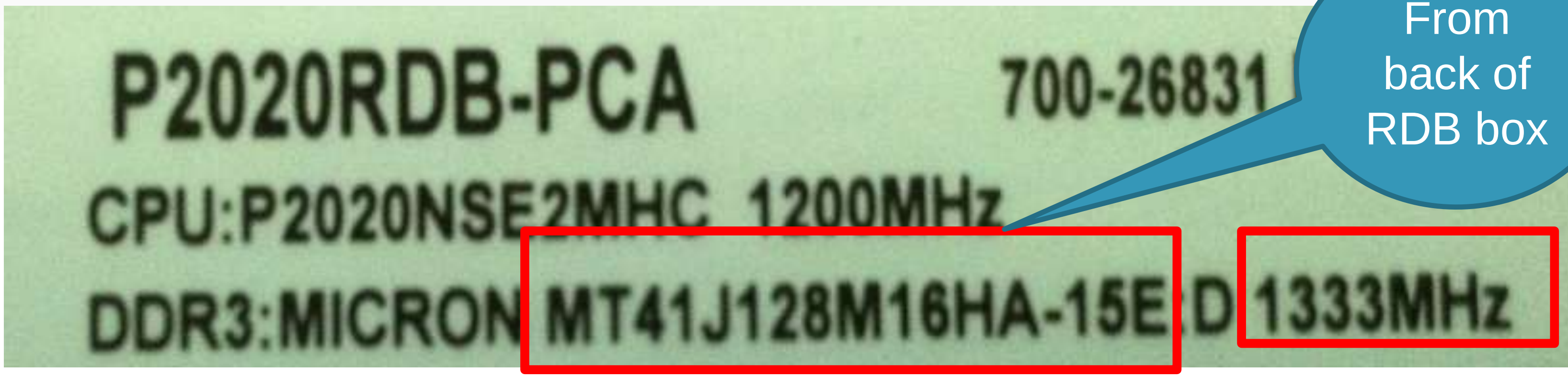


2

Change to 2Gb: 128Mb x 16

- From memory data sheet:
  - Maximum speed rating
  - Capacity

Get DRAM information – P2020RDB-PCA



From back of RDB box

Micron

DDR3 SDRAM

MT41J512M4 – 64 Meg x 4 x 8 Banks

MT41J256M8 – 32 Meg x 8 x 8 Banks

MT41J128M16 – 16 Meg x 16 x 8 Banks

From DRAM datasheet

freescale™

How about rest of the timing parameters?

Table 1: Key Timing Parameters

Speed Grade	Data Rate (MT/s)	Target tRCD-tRP-CL	tRCD (ns)	tRP (ns)	CL (ns)
-093[1, 2, 3, 4]	2133	14-14-14	13.09	13.09	13.09
-107[1, 2, 3]	1866	13-13-13	13.91	13.91	13.91
-125[1, 2,]	1600	11-11-11	13.75	13.75	13.75
-15E[1,]	1333	9-9-9	13.5	13.5	13.5
-187E	1066	7-7-7	13.1	13.1	13.1

Features

- $V_{DD} = V_{DDQ} = 1.5V \pm 0.075V$
- 1.5V center-terminated push/pull I/O
- Differential bidirectional data strobe
- 8*n*-bit prefetch architecture
- Differential clock inputs (CK, CK#)
- 8 internal banks
- Nominal and dynamic on-die termination (ODT) for data, strobe, and mask signals
- Programmable CAS READ latency (CL)
- Posted CAS additive latency (AL)
- Programmable CAS WRITE latency (CWL) based on tCK
- Fixed burst length (BL) of 8 and burst chop (BC) of 4 (via the mode register set [MRS])
- Selectable BC4 or BL8 on-the-fly (OTF)
- Self refresh mode
- T_C of 0°C to 95°C
  - 64ms, 8192 cycle refresh at 0°C to 85°C
  - 32ms, 8192 cycle refresh at 85°C to 95°C
- Self refresh temperature (SRT)
- Write leveling
- Multipurpose register
- Output driver calibration

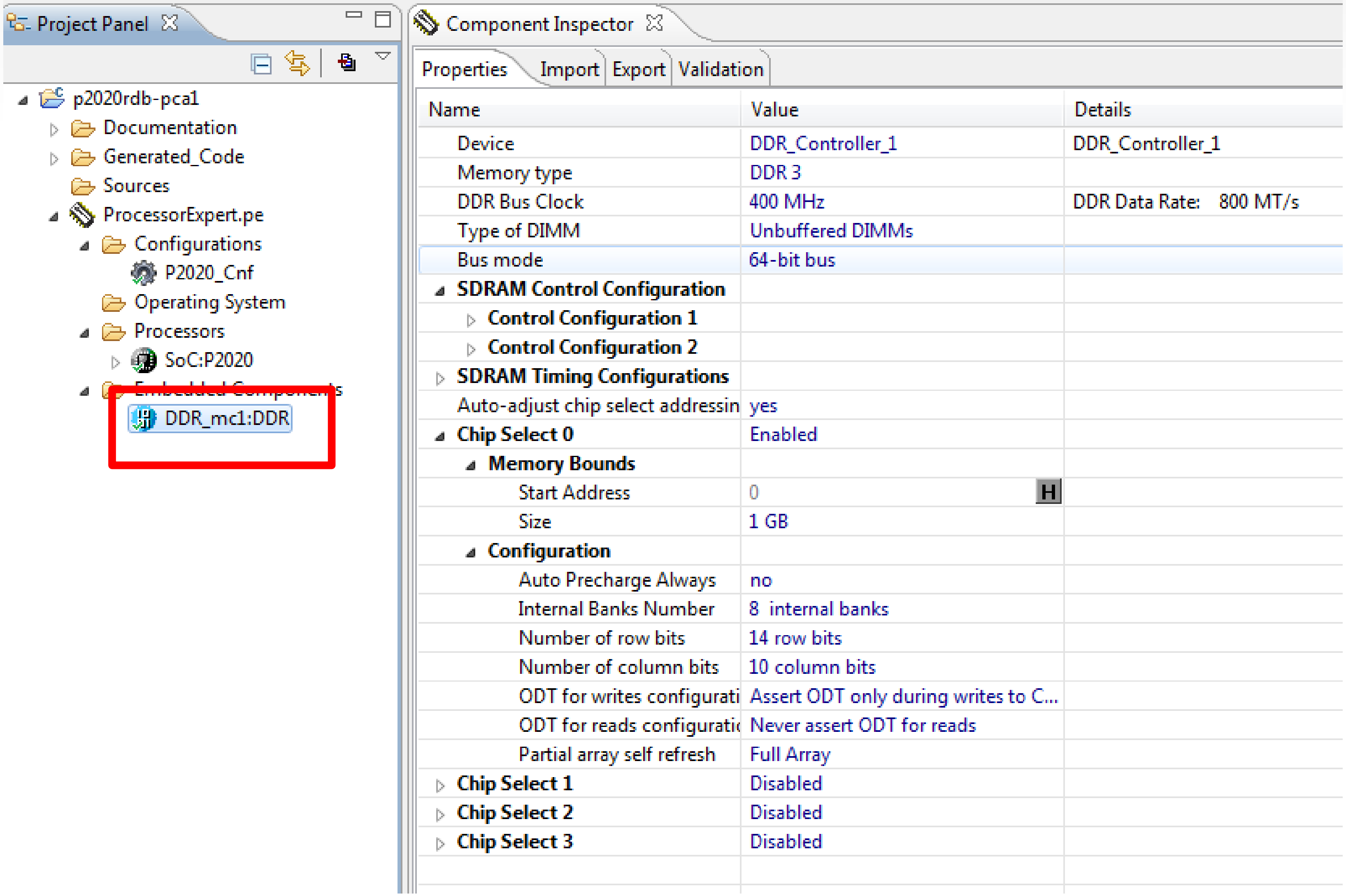
Options[1]	Marking
• Configuration	
– 512 Meg x 4	512M4
– 256 Meg x 8	256M8
– 128 Meg x 16	128M16
• FBGA package (Pb-free) – x4, x8	
– 78-ball (8mm x 10.5mm) Rev. H,M,J,K	DA
– 78-ball (9mm x 11.5mm) Rev. D	HX
• FBGA package (Pb-free) – x16	
– 96-ball (9mm x 14mm) Rev. D	HA
– 96-ball (8mm x 14mm) Rev. K	JT
• Timing – cycle time	
– 938ps @ CL = 14 (DDR3-2133)	-093
– 1.071ns @ CL = 13 (DDR3-1866)	-107
– 1.25ns @ CL = 11 (DDR3-1600)	-125
– 1.5ns @ CL = 9 (DDR3-1333)	-15E
– 1.87ns @ CL = 7 (DDR3-1066)	-187E
• Operating temperature	
– Commercial (0°C ≤ T_C ≤ +95°C)	None
– Industrial (–40°C ≤ T_C ≤ +95°C)	IT
• Revision	:D/:H/:J/:K/:M

Note: 1. Not all options listed can be combined to define an offered product. Use the part catalog search on http://www.micron.com for available offerings.

- Tool automatically computes tRCD, tRP, and CL!
  - User can change these values if required.

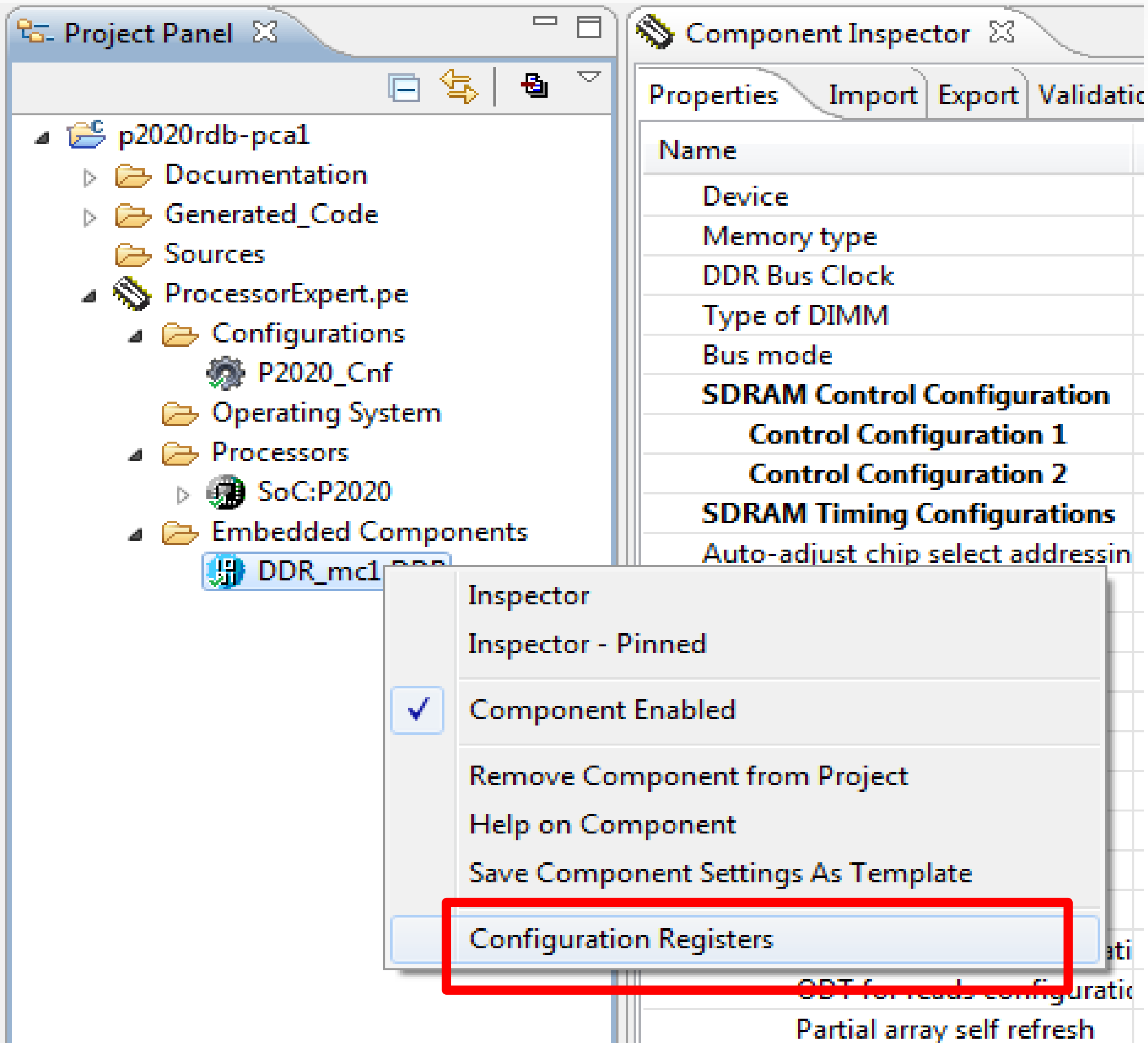
freescale™

QCS project explorer



freescale™

Review DDR registers values



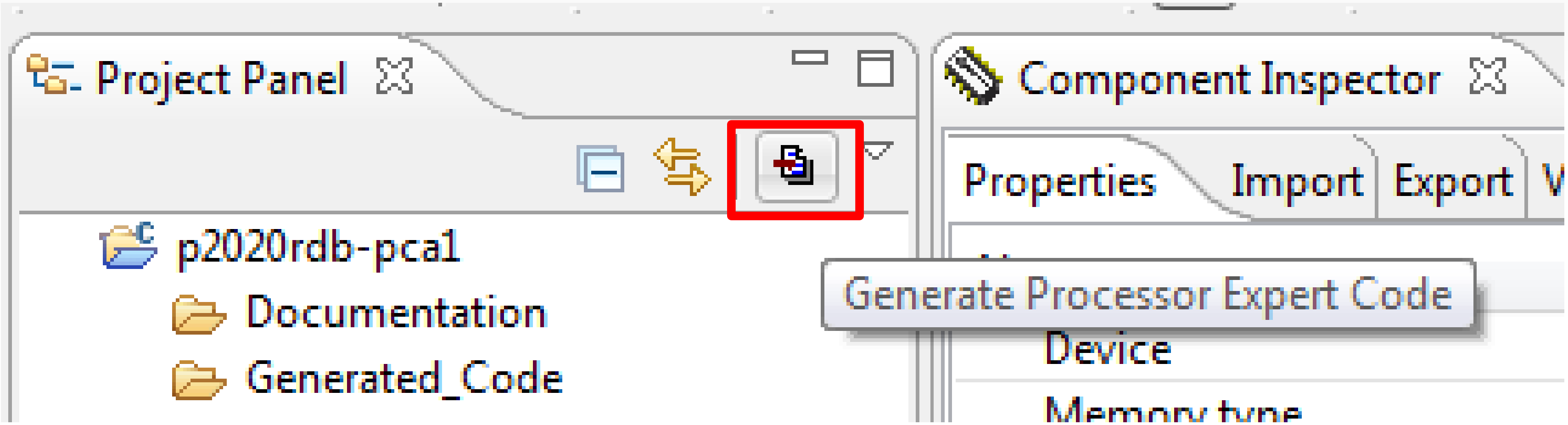
Review DDR registers values – contd.

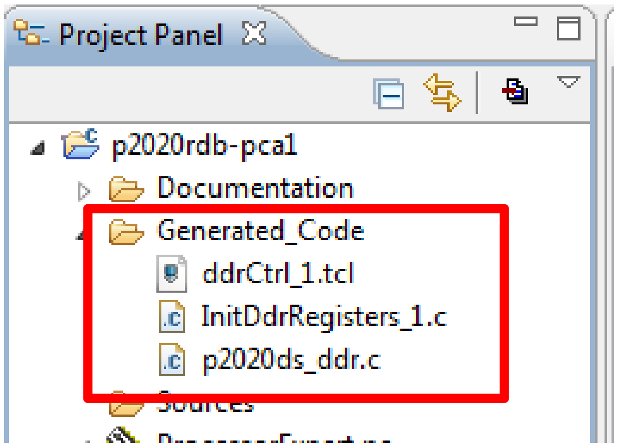
Configuration Registers

DDR_Controller_1

Reg. name	Init. value	After reset
Peripheral registers		
DDR1_CS0_BNDS	0000003F	00000000
DDR1_CS1_BNDS	00000000	00000000
DDR1_CS2_BNDS	00000000	00000000
DDR1_CS3_BNDS	00000000	00000000
DDR1_CS0_CONFIG	80014202	00000000
DDR1_CS1_CONFIG	00000000	00000000
DDR1_CS2_CONFIG	00000000	00000000
DDR1_CS3_CONFIG	00000000	00000000
DDR1_CS0_CONFIG_2	00000000	00000000
DDR1_CS1_CONFIG_2	00000000	00000000
DDR1_CS2_CONFIG_2	00000000	00000000
DDR1_CS3_CONFIG_2	00000000	00000000
DDR1_TIMING_CFG_3	00030000	00000000
DDR1_TIMING_CFG_0	00330104	00110105
DDR1_TIMING_CFG_1	6E6B8846	00000000
DDR1_TIMING_CFG_2	0FA8D0CC	00000000
DDR1_SDRAM_CFG	47000008	03000000
DDR1_SDRAM_CFG_2	24401050	00000000
DDR1_SDRAM_MODE	00061421	00000000

Generate DDR configuration





Generated files – CW, uboot, ddrinit.c

```
# DDR Controller 1 Registers

  # DDR_SDRAM_CFG
  mem [0xFF702110] = 0x47000008

  # CS0_BNDS
  mem [0xFF702000] = 0x3F

  # CS0_CONFIG
  mem [0xFF702080] = 0x80014202

  # CS0_CONFIG_2
  mem [0xFF7020C0] = 0x00

  # TIMING_CFG_3
  mem [0xFF702100] = 0x00030000

  # TIMING_CFG_0
  mem [0xFF702104] = 0x00330104

  # TIMING_CFG_1
  mem [0xFF702108] = 0x6E6B8846

  # TIMING_CFG_2
  mem [0xFF70210C] = 0x0FA8D0CC

  # DDR_SDRAM_CFG_2
  mem [0xFF702114] = 0x24401050

  # DDR_SDRAM_MODE
  mem [0xFF702118] = 0x00061421
```

```
#define DDR_1_INIT_EXT_ADDR_ADDR     0xFF70214C
#define DDR_1_SDRAM_RCW_1_ADDR       0xFF702180
#define DDR_1_SDRAM_RCW_2_ADDR       0xFF702184
#define DDR_1_DATA_INIT_ADDR         0xFF702128
#define DDR_1_SDRAM_MD_CNTL_ADDR     0xFF702120
#define DDR_1_DDRCDR_1_ADDR          0xFF702B28
#define DDR_1_DDRCDR_2_ADDR          0xFF702B2C

#define SDRAM_CFG_MEM_EN_MASK        0x80000000
#define SDRAM_CFG2_D_INIT_MASK       0x00000010

/* DDR Controller configured registers' values */
#define DDR_1_CS0_BNDS_VAL          0x3F
#define DDR_1_CS1_BNDS_VAL          0x00
#define DDR_1_CS2_BNDS_VAL          0x00
#define DDR_1_CS3_BNDS_VAL          0x00
#define DDR_1_CS0_CONFIG_VAL        0x80014202
#define DDR_1_CS1_CONFIG_VAL        0x00
```

```
#define PEX_CONFIG_DDR1_INIT_EXT_ADDR     0x00000000
#define PEX_CONFIG_DDR1_TIMING_4          0x00220001
#define PEX_CONFIG_DDR1_TIMING_5          0x02401400
#define PEX_CONFIG_DDR1_ZQ_CNTL           0x89080600
#define PEX_CONFIG_DDR1_WRLVL_CNTL        0x8655F614
#define PEX_CONFIG_DDR1_RCW_1             0x00000000
#define PEX_CONFIG_DDR1_RCW_2             0x00000000


/* DDR Controller 1 configuration global structures */
fsl_ddr_cfg_regs_t ddr_cfg_regs_0 = {
  .cs[0].bnds = PEX_CONFIG_DDR1_CS0_BNDS,
  .cs[1].bnds = PEX_CONFIG_DDR1_CS1_BNDS,
  .cs[2].bnds = PEX_CONFIG_DDR1_CS2_BNDS,
  .cs[3].bnds = PEX_CONFIG_DDR1_CS3_BNDS,
  .cs[0].config = PEX_CONFIG_DDR1_CS0_CONFIG,
  .cs[1].config = PEX_CONFIG_DDR1_CS1_CONFIG,
  .cs[2].config = PEX_CONFIG_DDR1_CS2_CONFIG,
```

Steps to adapt DDR configuration file in CodeWarrior

- Open the CW config file you want to adapt

  D:\Program Files\Freescale\CW PA v10.1\PA\PA_Support\Initialization_Files\QorIQ_P4\P4080DS_init_core0.cfg

- Replace DDR1 config section with the one from

  D:\Profiles\b08844\workspace\p4080\Generated_Code\ddrCtrl_1.cfg

- Use this new config file with your stationary project

DDR Configuration

Lab 4b: Changing the DDR Configuration - Starting with existing board values

DDR Step 1: Dump u-boot DDR Registers

- Deploy SDK2.3 u-boot
- No interleaving "fsl_ddr:ctlr_intlv=null"
- Use CW to connect and dump DDR1 and DDR2 registers (Memory browser export function)

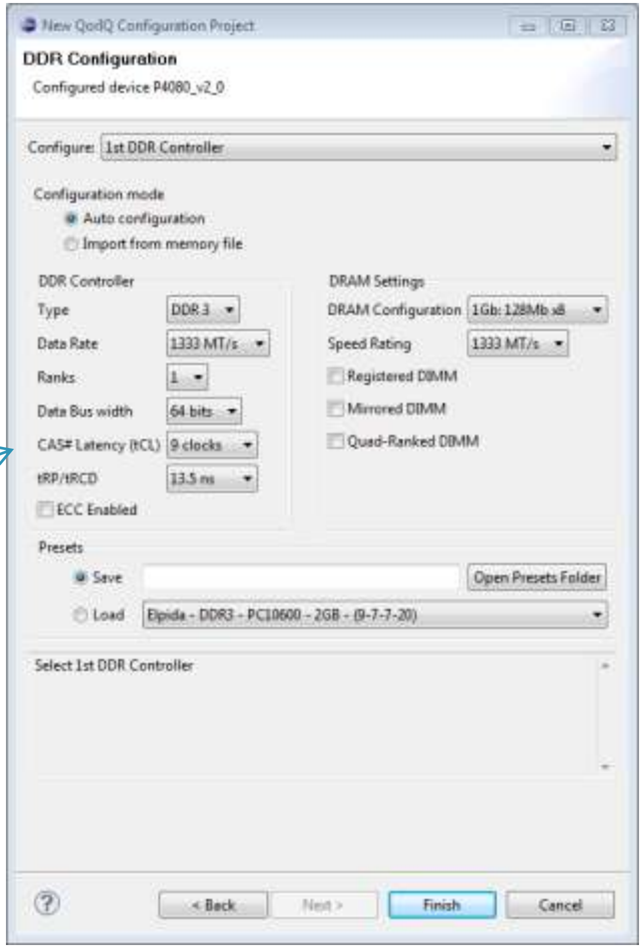
```
U-Boot 2010.12-00001-g612800e (Jan 28 2011 - 22:20:46)

CPU0:  P4080E, Version: 2.0, (0x82080020)
Core:  E500MC, Version: 2.0, (0x80230020)
Clock Configuration:
       CPU0:1499.985 MHz, CPU1:1499.985 MHz, CPU2:1499.985 MHz,
CPU3:1499.985 MHz,
       CPU4:1499.985 MHz, CPU5:1499.985 MHz, CPU6:1499.985 MHz,
CPU7:1499.985 MHz,
       CCB:799.992 MHz,
       DDR:649.994 MHz (1299.987 MT/s data rate) (Asynchronous),
LBC:99.999 MHz
       FMAN1: 599.994 MHz
       FMAN2: 599.994 MHz
       PME:   399.996 MHz
L1:    D-cache 32 kB enabled
       I-cache 32 kB enabled
Board: P4080DS, Sys ID: 0x17, Sys Ver: 0x01, FPGA Ver: 0x0c, vBank: 4
36-bit Addressing
Reset Configuration Word (RCW):
       00000000: 105a0000 00000000 1e1e181e 0000cccc
       00000010: 40464000 3c3c2000 fe800000 61000000
       00000020: 00000000 00000000 00000000 008b6000
       00000030: 00000000 00000000 00000000 00000000
SERDES Reference Clocks: Bank1=125MHz Bank2=125MHz Bank3=125MHz
I2C:   ready
DRAM:  Initializing....using SPD
Detected UDIMM(s)
Detected UDIMM(s)
2 GiB left unmapped
   DDR: 4 GiB (DDR3, 64-bit, CL=9, ECC on)
Testing 0x00000000 - 0x7fffffff
Testing 0x80000000 - 0xffffffff
Remap DDR 2 GiB left unmapped

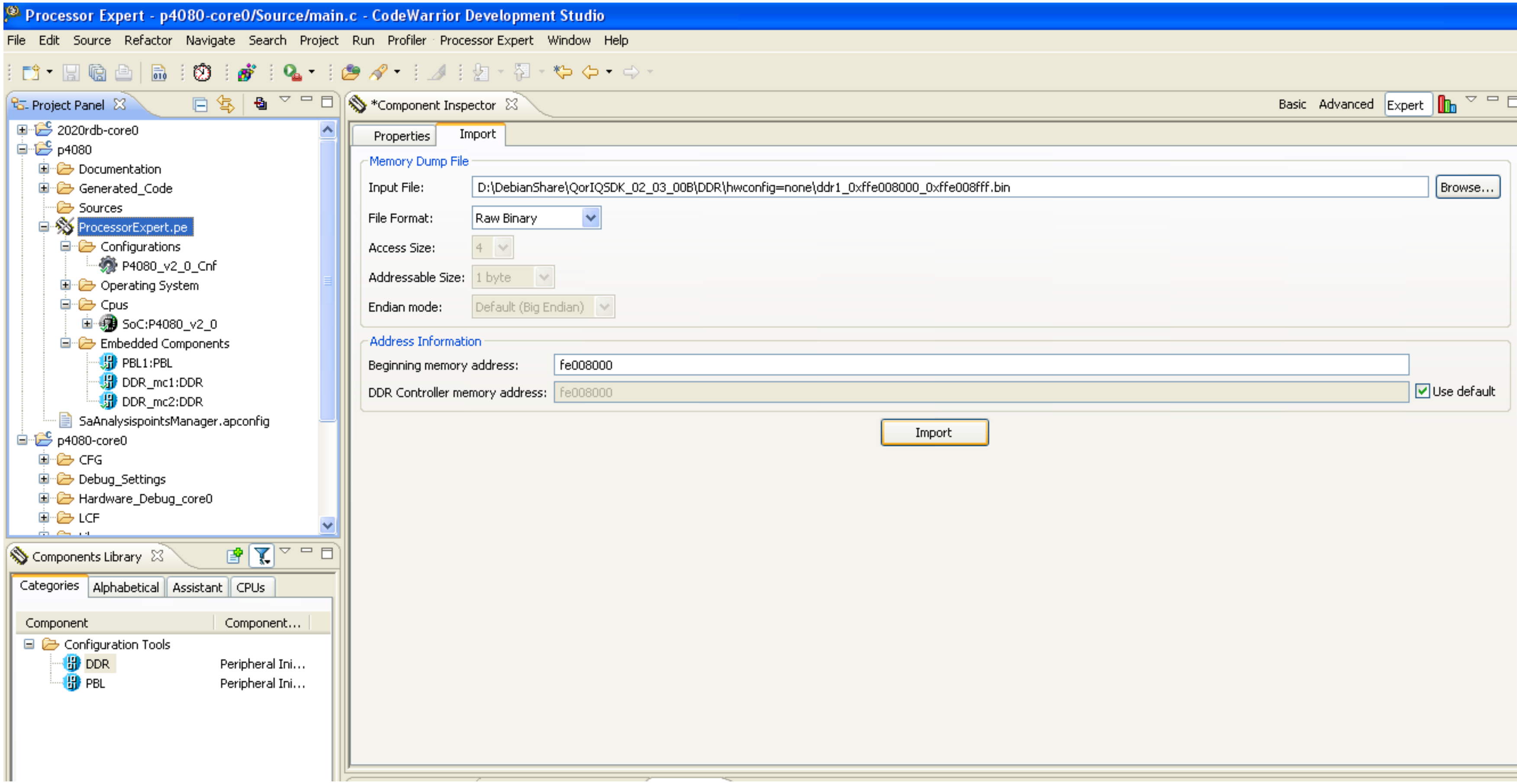
Hit any key to stop autoboot:  0
=> md 0xfe008000
```

Create Lab4 project

- File > New > QorIQ Configuration Project
- Select P4080 rev 2
- Select only DDR tool
- Take the default configuration



DDR Step 2: Import DDR1 and DDR2 Registers Dump Under QCS



freescale™

DDR Step 3: Review Decoded Configurations

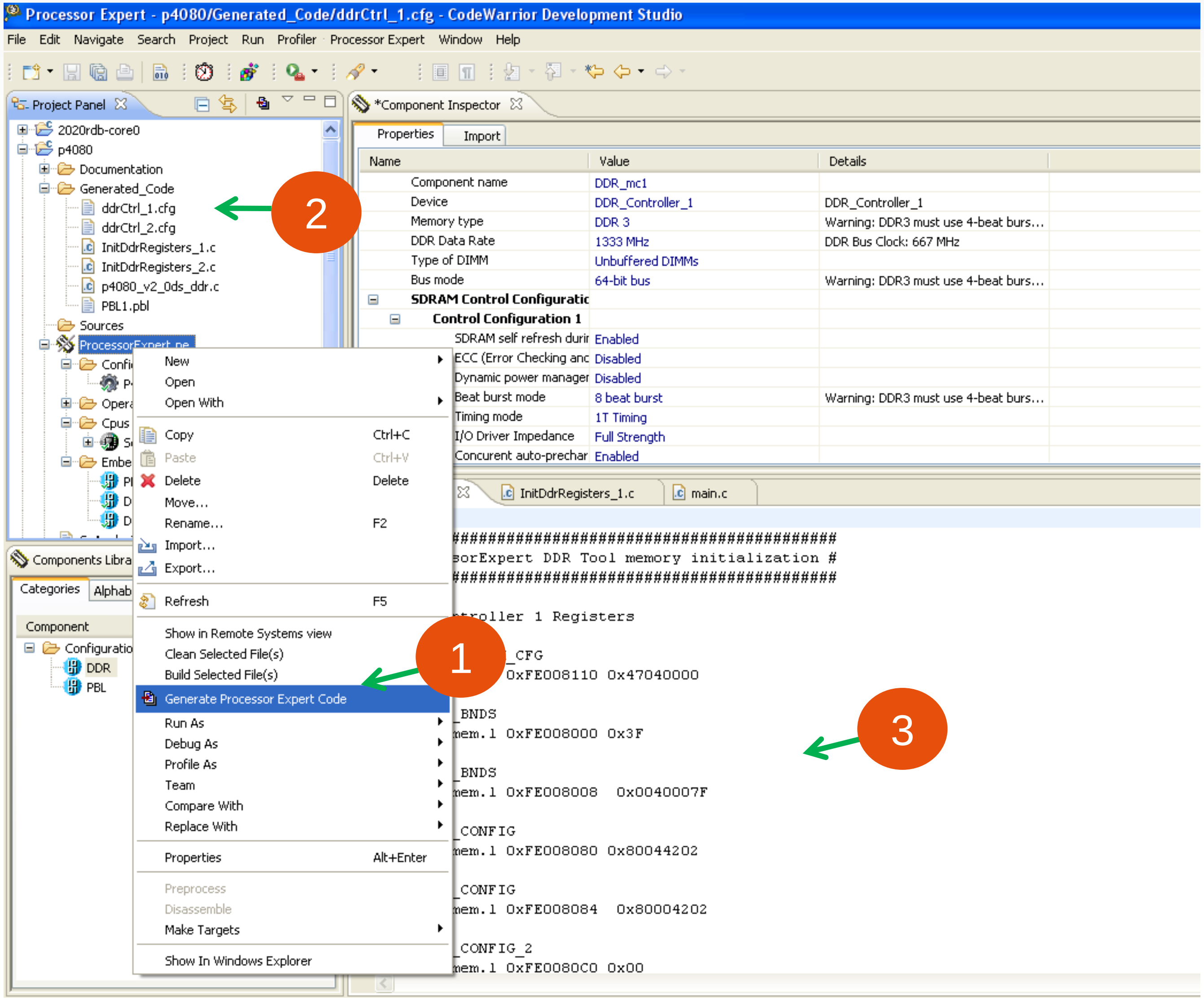
*Component Inspector — Basic | Advanced | Expert

Properties | Import | Export

Name	Value	Details
Device	DDR_Controller_1	DDR_Controller_1
Memory type	DDR 3	
DDR Bus Clock	667 MHz	DDR Data Rate: 1334 MT/s
Type of DIMM	Unbuffered DIMMs	
Bus mode	64-bit bus	
SDRAM Control Configuration		
Control Configuration 1		
SDRAM self refresh during	Enabled	
ECC (Error Checking and (	Enabled	
Dynamic power managem	Disabled	
Beat burst mode	8 beat burst	
Timing mode	1T Timing	
I/O Driver Impedance	Full Strength	
Concurent auto-precharg	Enabled	
Control Configuration 2		
DLL Reset	no	
DQS Configuration	Use differential DQS signals	
ODT Configuration	Assert ODT to internal IOs only dur...	
Number of posted refresh	1 refresh is issued at a time	
Use quad-ranked DIMM	no	
Address Parity	Disabled	
Register Control Word	Disabled	
Corrupted data feature	Enabled	
Use mirrored DIMMs	yes	
SDRAM Timing Configurations		
Auto-adjust chip select addressin	no	
Chip Select 0	Enabled	
Memory Bounds		
Start Address	0	

freescale™

DDR Step 4: Generate DDR Config File



DDR Step 5: Adapt CW Config File

- **Open the CW config file you want to adapt**

  D:\Program Files\Freescale\CW PA v10.1\PA\PA_Support\Initialization_Files\QorIQ_P4\P4080DS_init_core0.tcl

- **Replace DDR1 config section with the one from:**

  D:\Profiles\b08844\workspace\p4080\Generated_Code\ddrCtrl_1.tcl

- **Use the new config file with your stationary project**

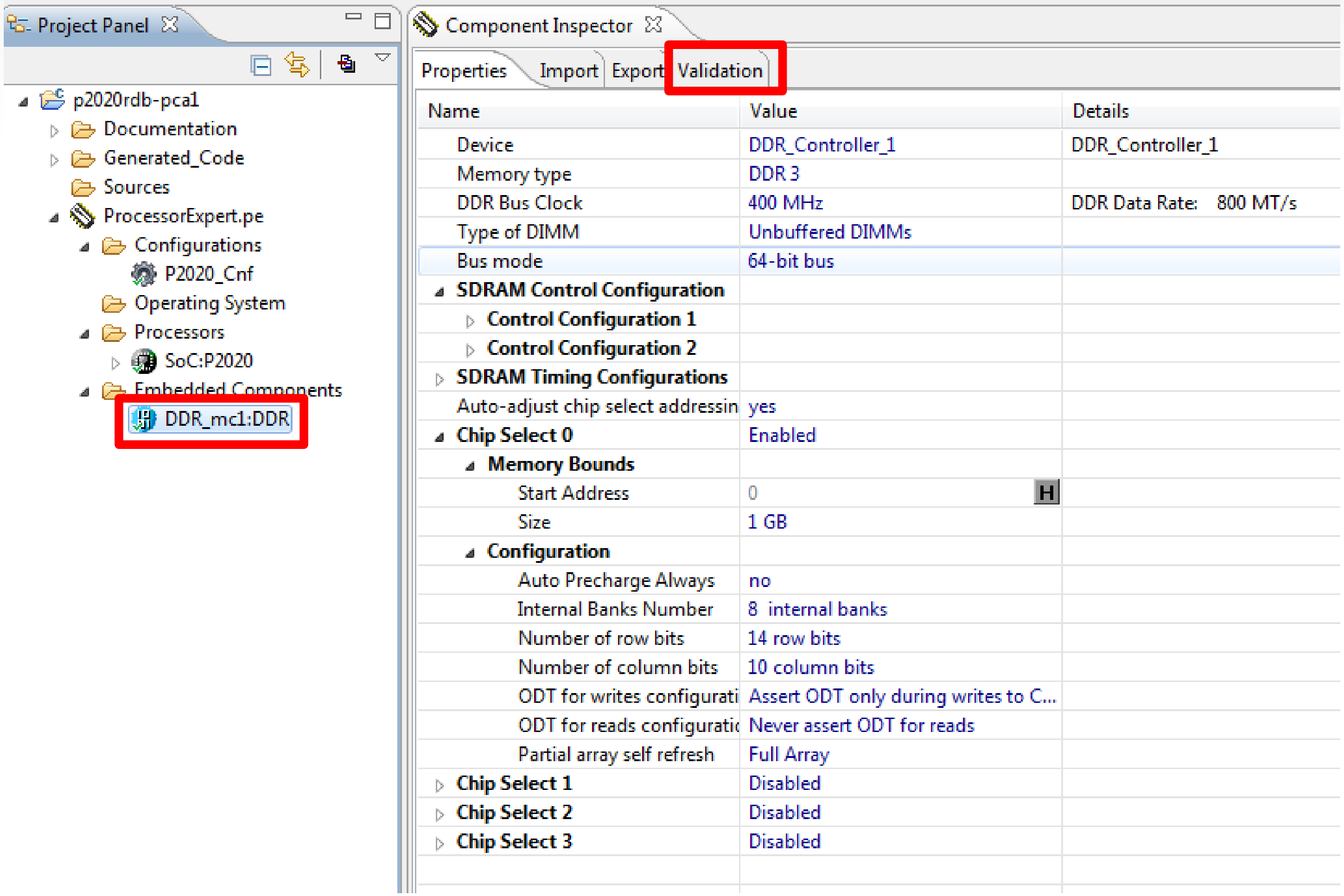
DDR Validation Tool (DDRv)

Lab 4c: Validating DDR Configuration

NOTES:

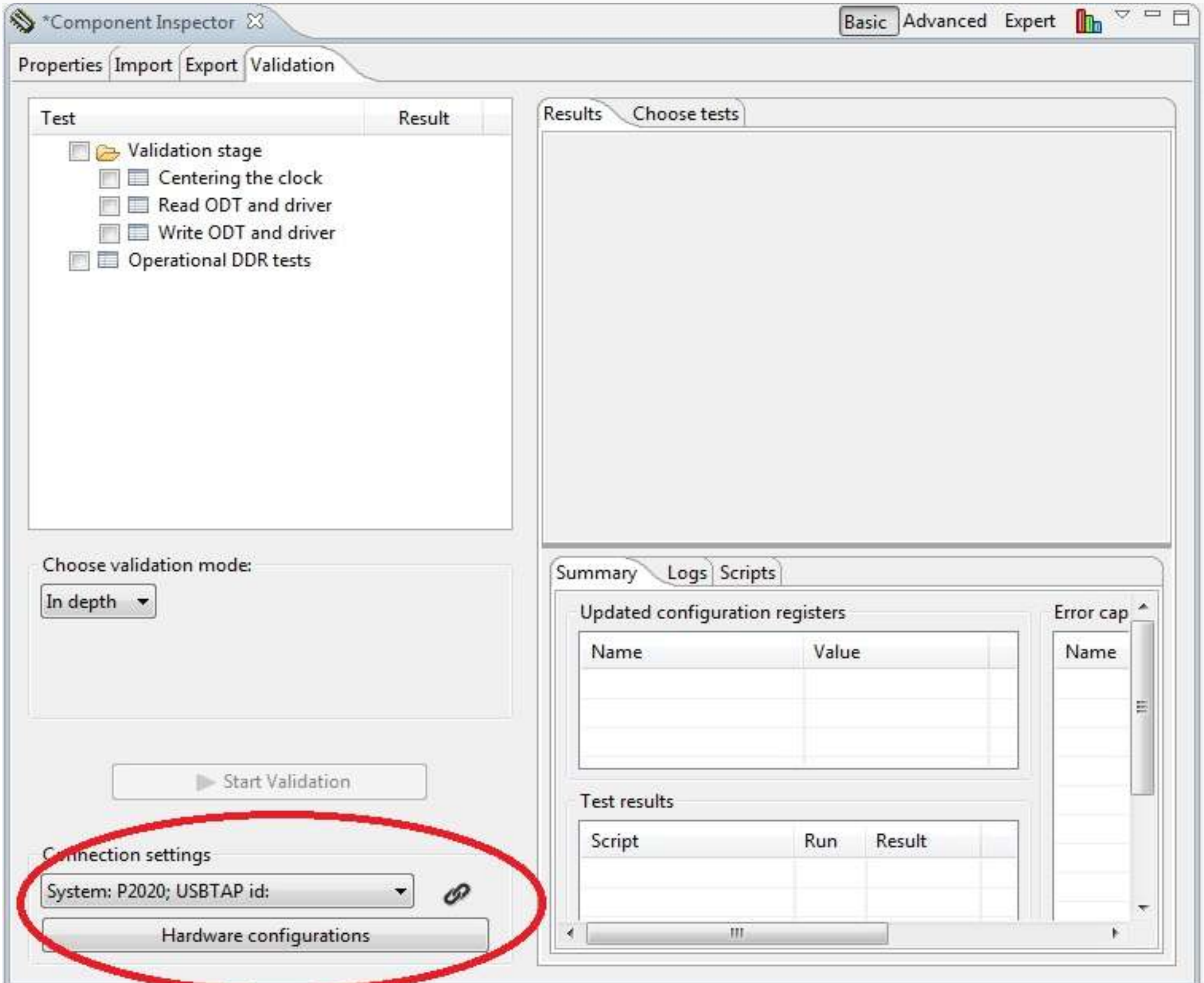
1. THIS TOOL IS AN ADD-IN to QCS
2. This tool is 995$ and requires a TAP to be purchased separately.
3. Current version is v1.3 and is available from the Freescale website – use Buy Direct to purchase online.

DDR Validation Tool - Activate

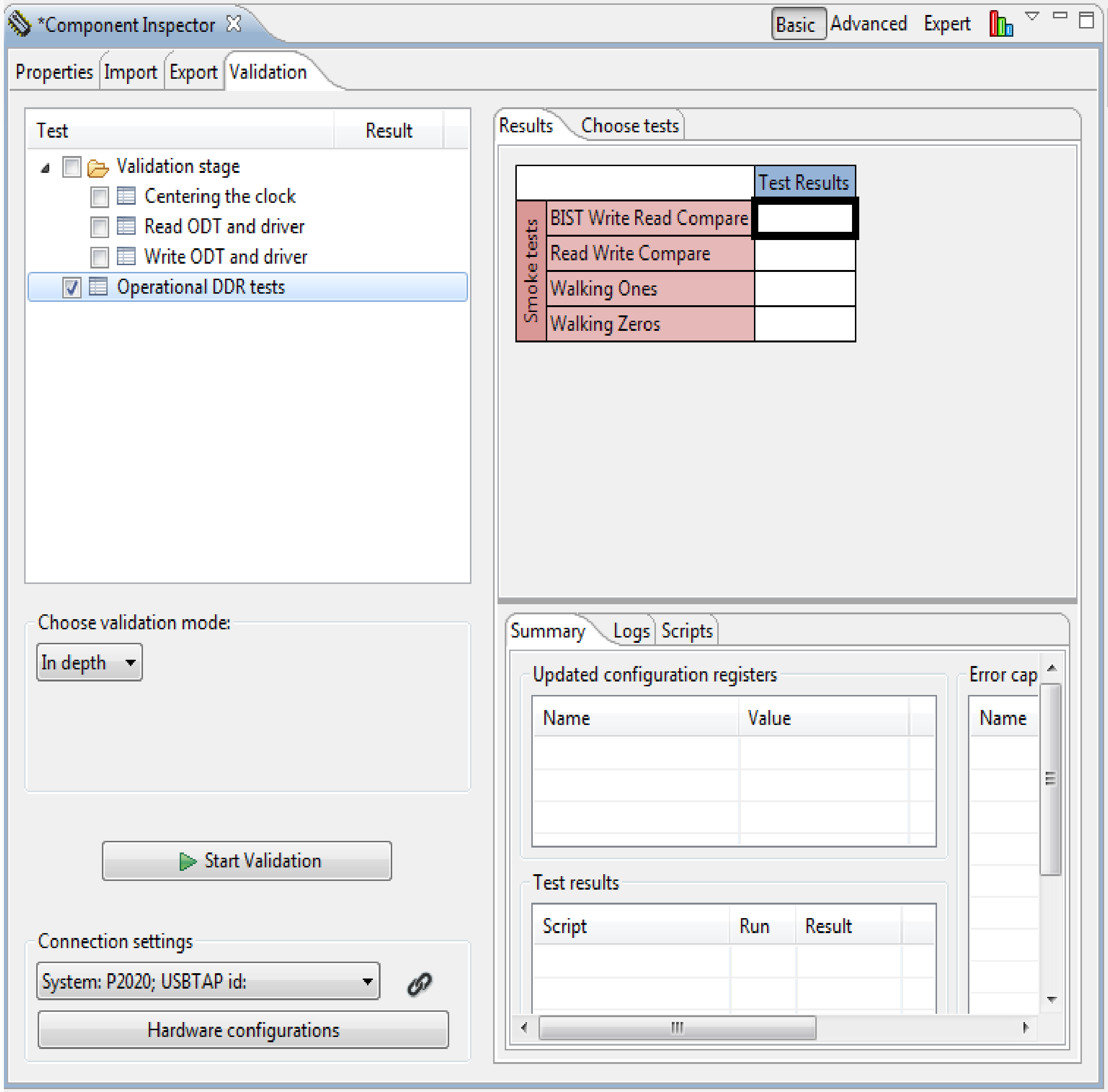


License file:
<QCS Install directory>/eclipse/Optimization/license.dat

DDRv Connection Setup

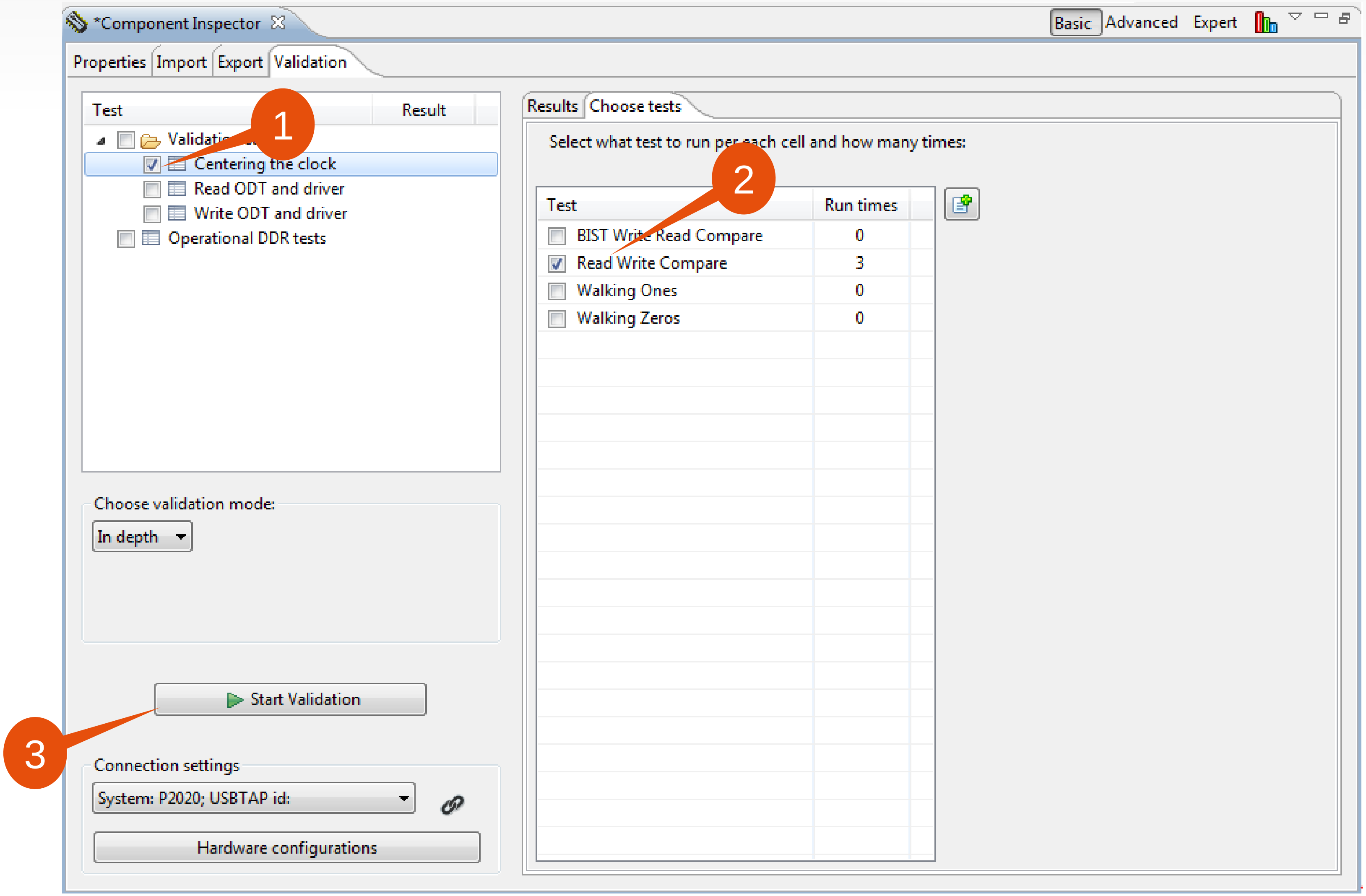


DDRv Basic Connection Test

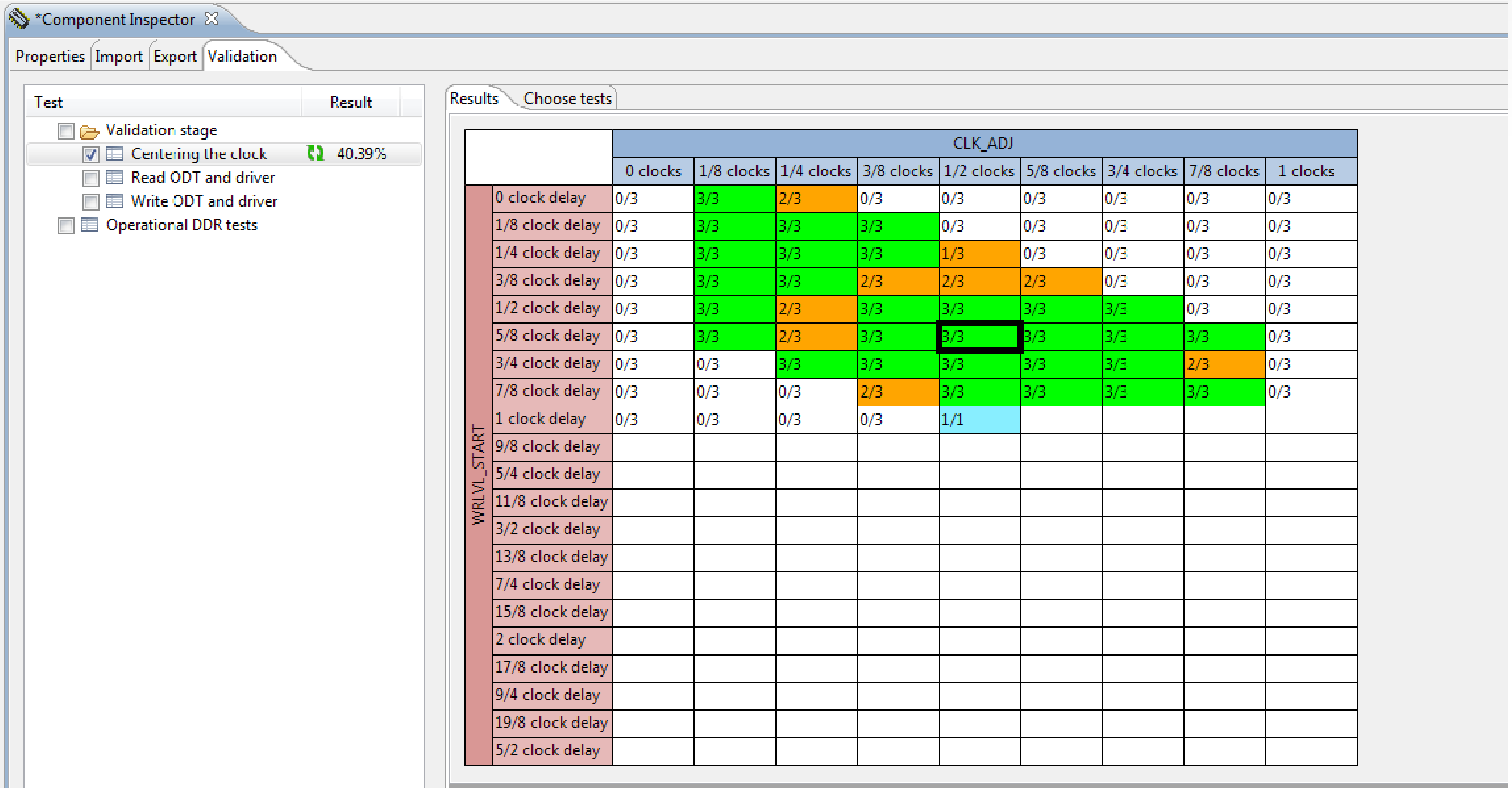


- Run basic test to confirm target connection

Configure DDR scenarios and tests



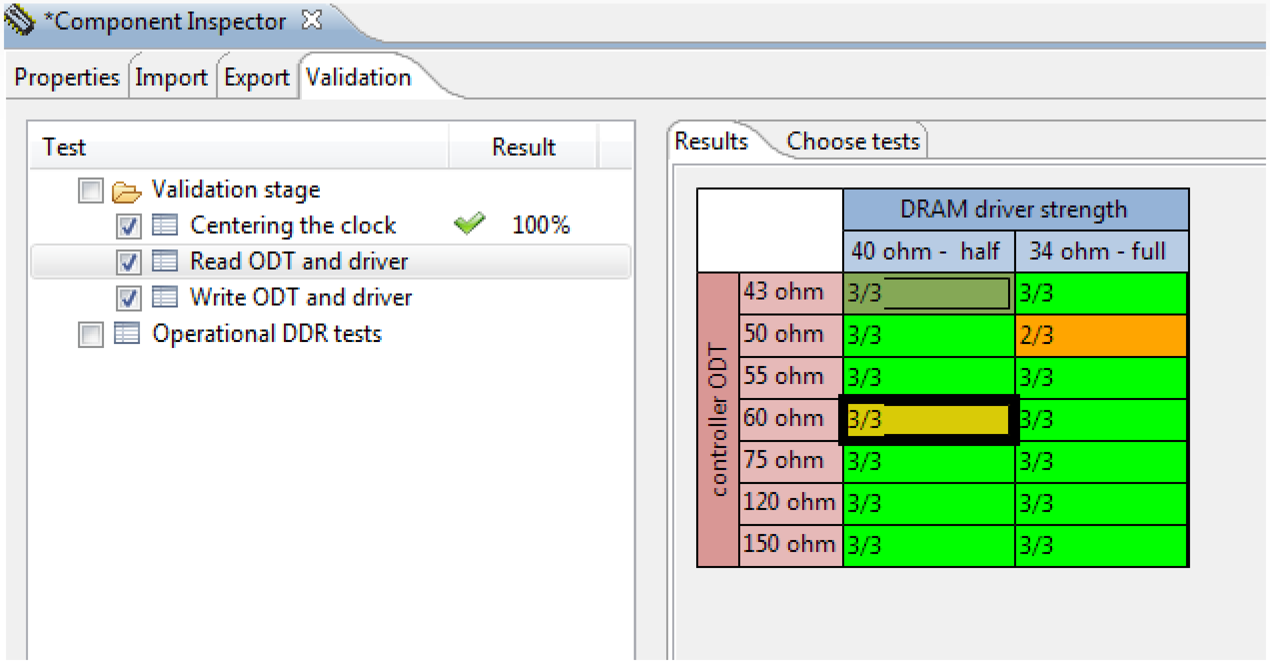
Centering of the clock results



	0 clocks	1/8 clocks	1/4 clocks	3/8 clocks	1/2 clocks	5/8 clocks	3/4 clocks	7/8 clocks	1 clocks
0 clock delay	0/3	3/3	2/3	0/3	0/3	0/3	0/3	0/3	0/3
1/8 clock delay	0/3	3/3	3/3	3/3	0/3	0/3	0/3	0/3	0/3
1/4 clock delay	0/3	3/3	3/3	3/3	1/3	0/3	0/3	0/3	0/3
3/8 clock delay	0/3	3/3	3/3	2/3	2/3	2/3	0/3	0/3	0/3
1/2 clock delay	0/3	3/3	2/3	3/3	3/3	3/3	3/3	0/3	0/3
5/8 clock delay	0/3	3/3	2/3	3/3	3/3	3/3	3/3	3/3	0/3
3/4 clock delay	0/3	0/3	3/3	3/3	3/3	3/3	3/3	2/3	0/3
7/8 clock delay	0/3	0/3	0/3	2/3	3/3	3/3	3/3	3/3	0/3
1 clock delay	0/3	0/3	0/3	0/3	1/1				
9/8 clock delay									
5/4 clock delay									
11/8 clock delay									
3/2 clock delay									
13/8 clock delay									
7/4 clock delay									
15/8 clock delay									
2 clock delay									
17/8 clock delay									
9/4 clock delay									
19/8 clock delay									
5/2 clock delay									

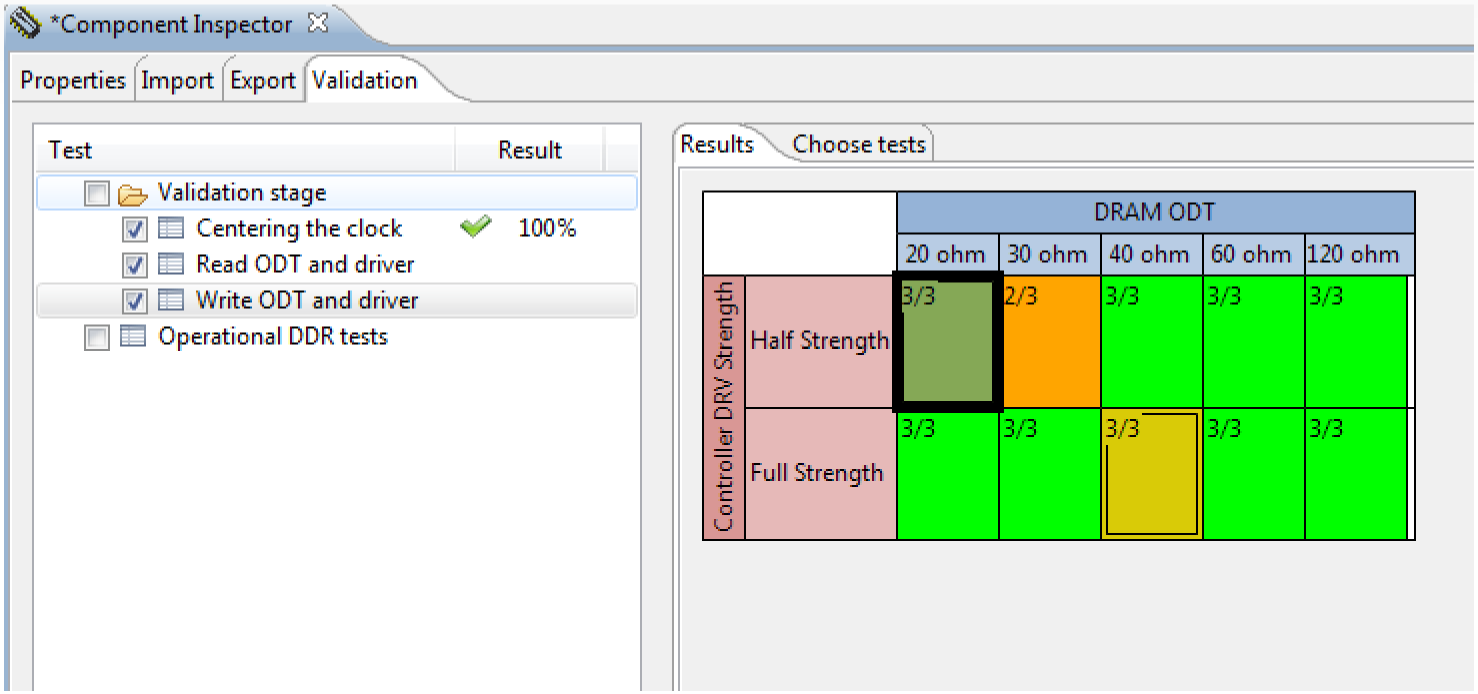
- Click “cell” to choose Write level start and CLK_ADJ values.

DDR read ODT and driver strength – test results



- Click “cell” to choose optimized ODT value.

DDR write ODT and drive strength – test results



- Click “cell” to choose optimized ODT value.

freescale™

Centering of the clock - after ODT optimization

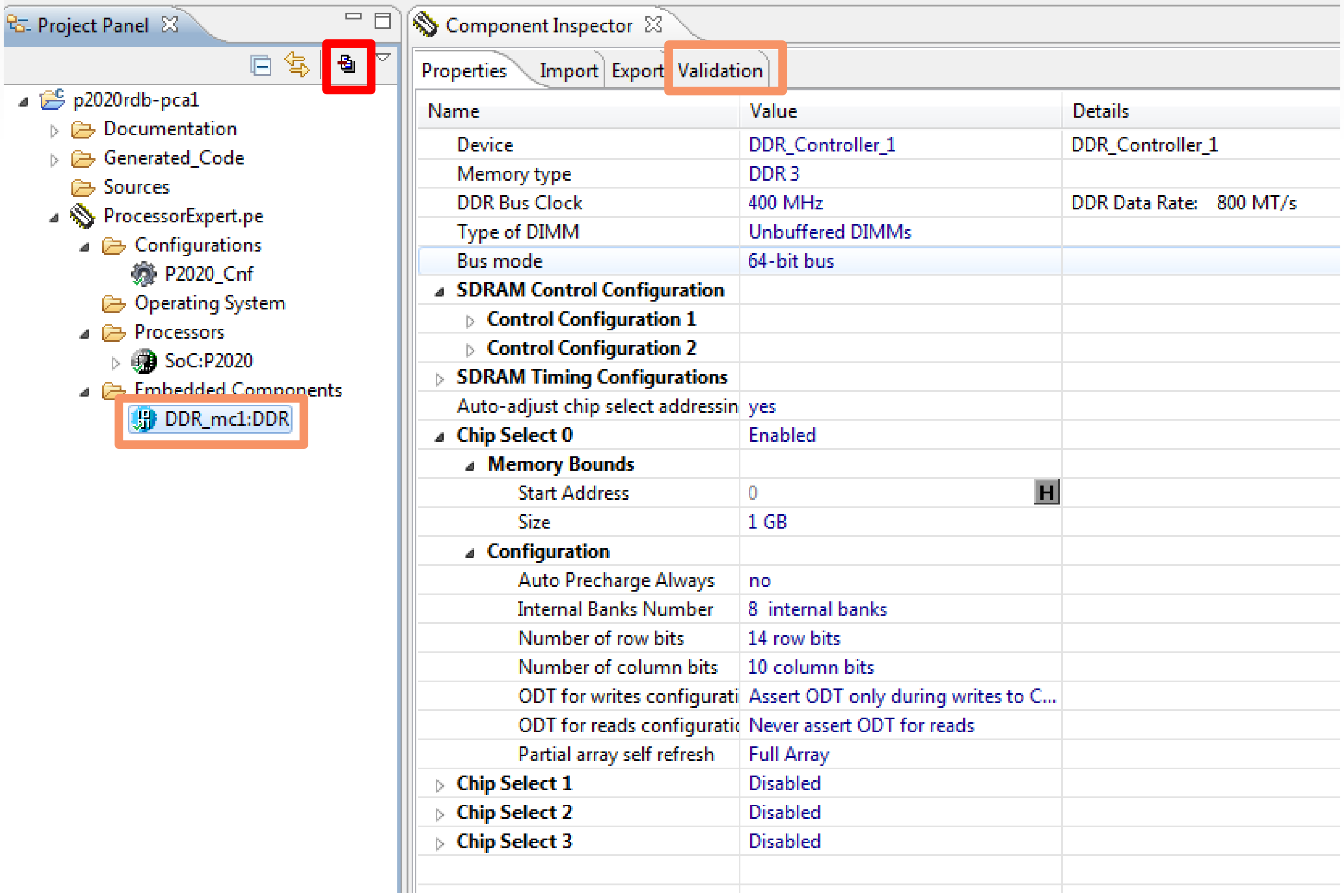
Results | Choose tests

WRLVL_START \ CLK_ADJ	0 clocks	1/8 clocks	1/4 clocks	3/8 clocks	1/2 clocks	5/8 clocks	3/4 clocks	7/8 clocks	1 clocks
0 clock delay	0/3	3/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3
1/8 clock delay	0/3	3/3	3/3	3/3	0/3	0/3	0/3	0/3	0/3
1/4 clock delay	0/3	3/3	3/3	3/3	0/3	0/3	0/3	0/3	0/3
3/8 clock delay	0/3	3/3	3/3	3/3	3/3	3/3	0/3	0/3	0/3
1/2 clock delay	0/3	3/3	3/3	3/3	3/3	3/3	0/3	0/3	0/3
5/8 clock delay	0/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	0/3
3/4 clock delay	0/3	0/3	3/3	3/3	3/3	3/3	3/3	3/3	0/3
7/8 clock delay	0/3	0/3	0/3	3/3	3/3	3/3	2/3	3/3	0/3
1 clock delay	0/3	0/3	0/3	0/3	3/3	3/3	3/3	3/3	0/3
9/8 clock delay	0/3	0/3	0/3	0/3	0/3	3/3	3/3	3/3	0/3
5/4 clock delay	0/3	0/3	0/3	0/3	0/3	0/3	2/3	3/3	0/3
11/8 clock delay	0/3	0/3	0/3	0/3	0/3	0/3	0/3	3/3	0/3
3/2 clock delay	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3
13/8 clock delay	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3
7/4 clock delay	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3
15/8 clock delay	0/3	3/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3
2 clock delay	0/3	3/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3
17/8 clock delay	0/3	3/3	3/3	3/3	0/3	0/3	0/3	0/3	0/3
9/4 clock delay	0/3	3/3	3/3	3/3	0/3	0/3	0/3	0/3	0/3
19/8 clock delay	0/3	3/3	3/3	3/3	3/3	3/3	0/3	0/3	0/3
5/2 clock delay	0/3	3/3	3/3	3/3	3/3	3/3	0/3	0/3	0/3

- Centering of clock scenario was re-run after finding the right ODT values

freescale™

Generate optimized DDR configuration



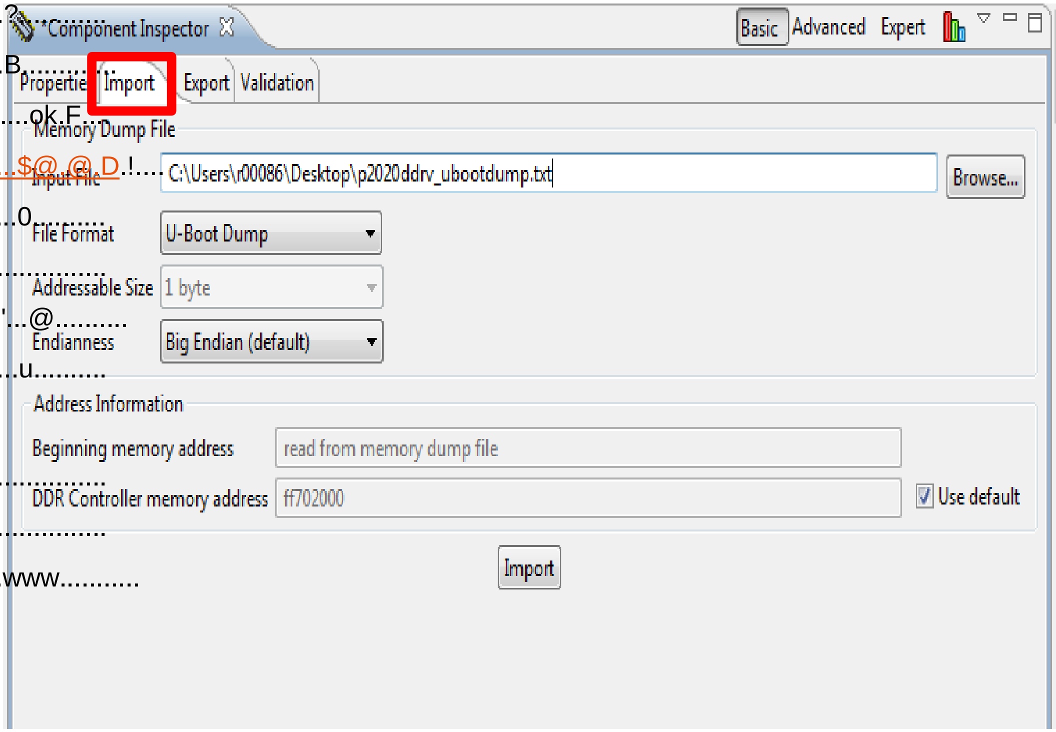
Pricing $995

License file:
<QCS Install directory>/eclipse/Optimization/license.dat

P2020RDB-PCA: Import DDR configuration from existing system running uboot

- At uboot prompt

- => md ffe02000
  - ffe02000: 0000003f 00000000 00000000 00000000 ...?
  - ffe02080: 80014202 00000000 00000000 00000000 ..B.........
  - ffe02100: 00030000 00110104 6f6b8846 0fa8c8cc ....ok.F....
  - ffe02110: c7000008 24401040 00441421 00000000 ....$@.@.D.!....
  - ffe02120: 00000000 0c300100 deadbeef 00000000 .....0..........
  - ffe02130: 03000000 00000000 00000000 00000000 ....
  - ffe02160: 00220001 02401400 00000000 00000000 ."...@..........
  - ffe02170: 89080600 8675f608 00000000 00000000 .....u..........
- => md ffe02b00
  - ffe02b00: 00000000 00000000 00000000 00000000 ....
  - ffe02b10: 00000000 00000000 00000000 00000000 ....
  - ffe02b20: 5dc07777 77000000 00000000 00000000].www...........



- Save content to a file.

Device Tree Editor

Lab 5: Changing the Hardware Device Tree

Device Tree Hands-on

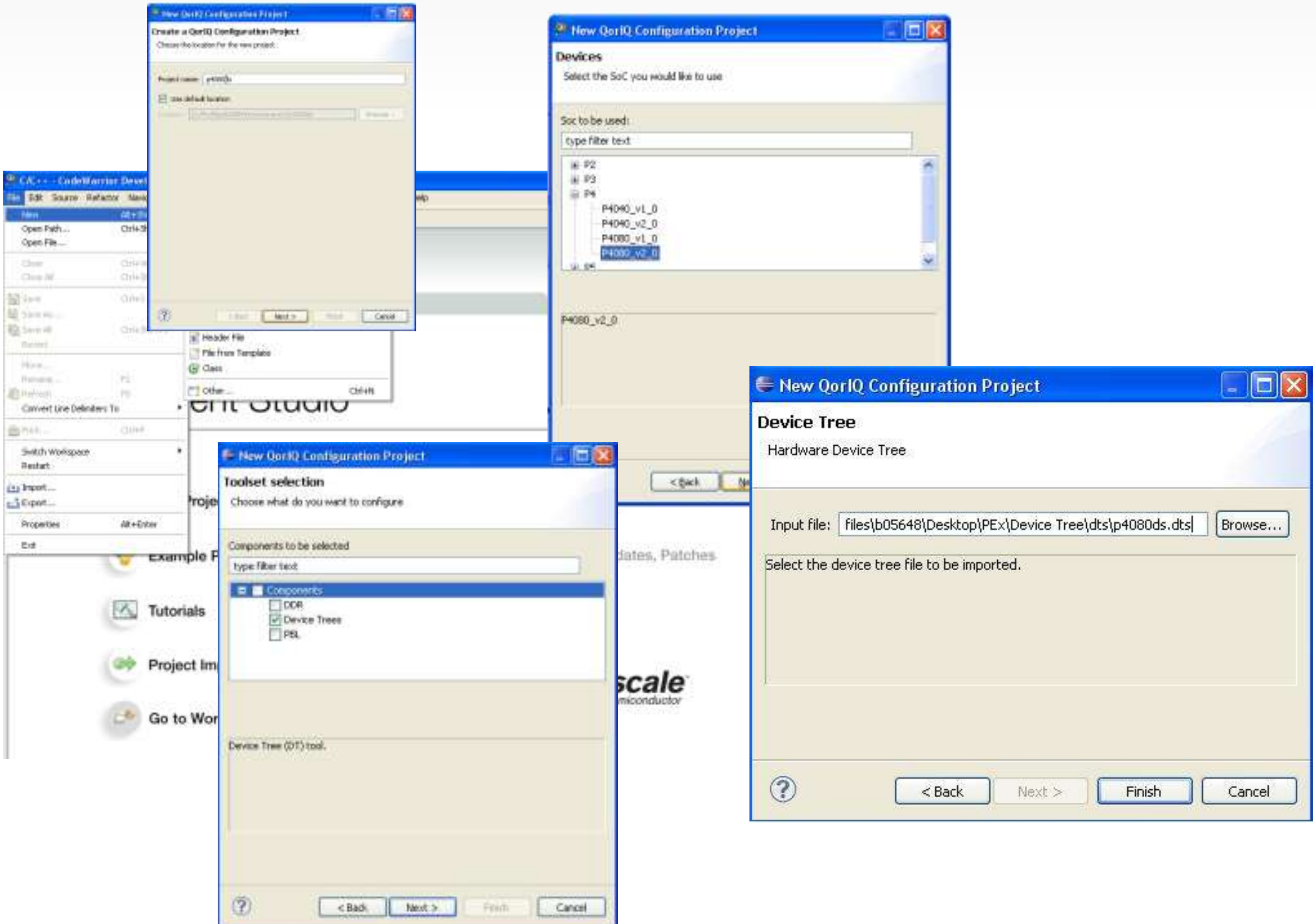
What we will do:

- Define hardware device tree for P4040 starting from P4080 device tree
  - Import the P4080 (which has 8 cores)
  - Configure to P4040 (which has 4 cores)
  - Note P4080 and P4040 are same SOC otherwise

- Walk through the next slides using QCS Hardware Device Tree editor to solve this scenario

freescale™

Device Tree Step 1: Create New Project



freescale™

Device Tree Step 1: Create New Project (cont.)

1. Steps:
   - File -> New -> QorIQ Configuration Project
   - enter Project name -> Next
   - select SoC (p4080_v2_0) -> Next
   - select «Device Trees» component -> Next
   - browse to an existing p4080ds.dts file -> Finish

2. Wait while the device tree component is updating with the imported data

3. Expand *ProcessorExpert.pe* -> *Embedded Components* -> click on *DT1: HWDeviceTree* component

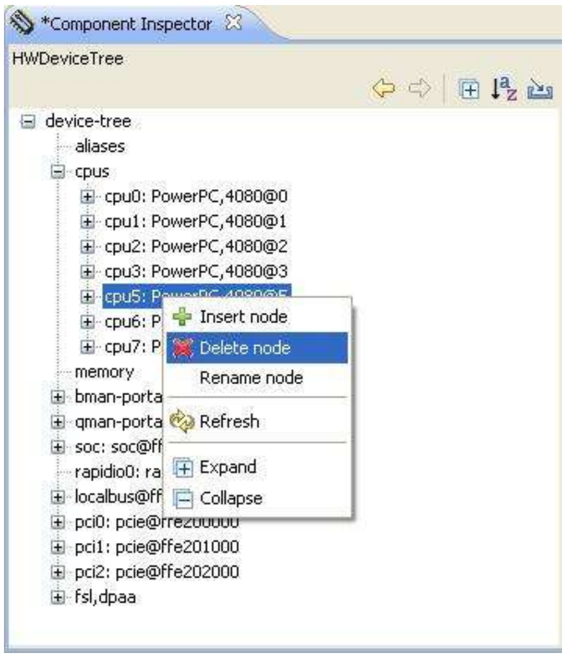
4. The imported dts file is added under *Imported_Files* folder

5. Open generated device tree file *Generated_Code/P4080_v2_0.dts*. At this moment the imported dts and the generated one are identical

Device Tree Step 2: Remove Unnecessary Nodes

6. Search for CPU nodes in the tree view; you should see 8 cores with 3 properties each
7. Delete cpu4-cpu7, you have two options:

- *Using graphical editor*



Delete one node at a time.
After each deletion, dts file is automatically generated.

OR

- *Using text editor*



Select all 4 nodes and press delete.
To reflect the modifications in the graphical editor too, save the file (Ctrl+S).

freescale™

Device Tree Step 3: Solve Validation Errors

8. Look in the Properties view - There are 8 errors
9. Click on each error and go to the corresponding line in the device tree file
10. There are undefined references in some nodes pointing to the removed cpus:

    bman-portal@10000

    bman-portal@14000

    bman-portal@18000

    bman-portal@1c000

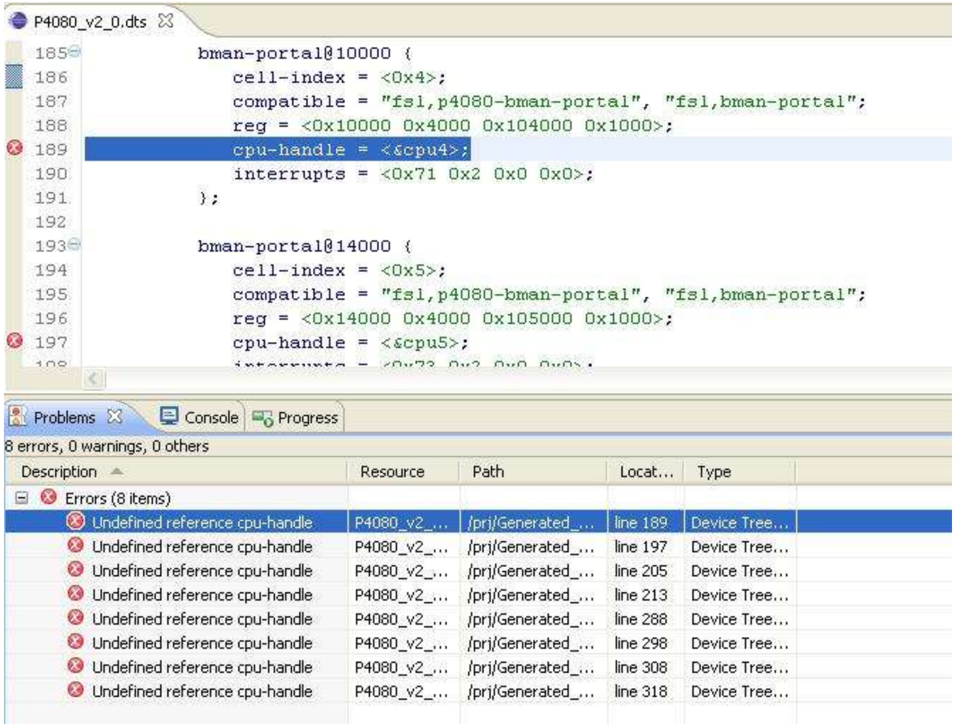
    qportal4

    qportal5

    qportal6

    qportal7

8. Remove the above nodes in a similar manner
9. Save your changes

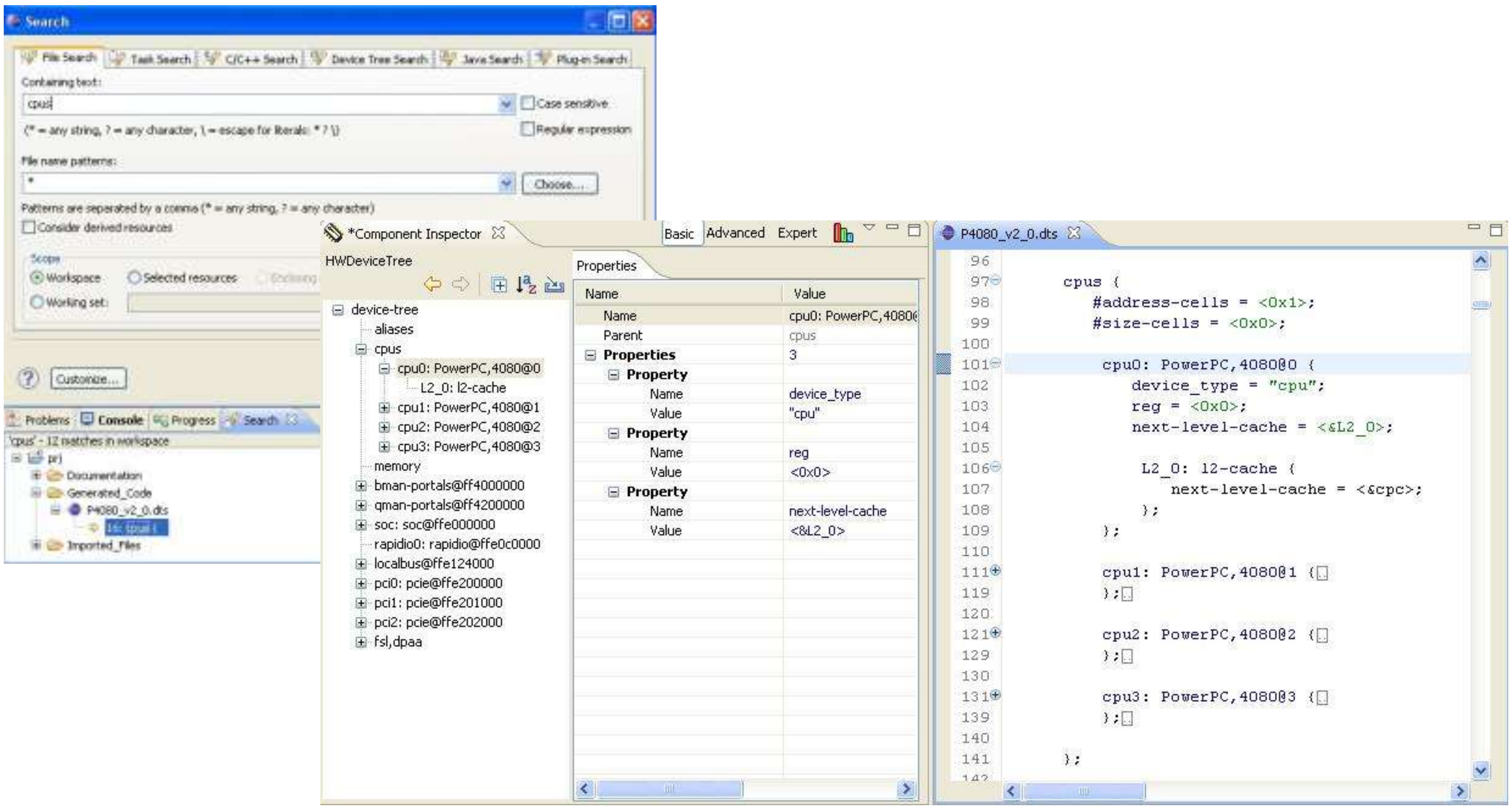


freescale™

Device Tree Step 4: P4040 Device Tree

13. Go to *Search* menu -> select *Device Tree Search* tab -> enter *cpus* text -> press *Search*
14. In *Search* view select the found matches associated with the generated file

- New device tree has 4 cores and looks as follows:



Device Tree Step 5: Apply and Test Changes

```
processor	: 0
cpu		: e500mc
clock		: 1499.985000MHz
revision	: 2.0 (pvr 8023 0020)
bogomips	: 99.99

processor	: 1
cpu		: e500mc
clock		: 1499.985000MHz
revision	: 2.0 (pvr 8023 0020)
bogomips	: 99.99

processor	: 2
cpu		: e500mc
clock		: 1499.985000MHz
revision	: 2.0 (pvr 8023 0020)
bogomips	: 99.99

processor	: 3
cpu		: e500mc
clock		: 1499.985000MHz
revision	: 2.0 (pvr 8023 0020)
bogomips	: 99.99

processor	: 4
cpu		: e500mc
clock		: 1499.985000MHz
revision	: 2.0 (pvr 8023 0020)
bogomips	: 99.99

processor	: 5
cpu		: e500mc
clock		: 1499.985000MHz
revision	: 2.0 (pvr 8023 0020)
bogomips	: 99.99

processor	: 6
cpu		: e500mc
clock		: 1499.985000MHz
revision	: 2.0 (pvr 8023 0020)
bogomips	: 99.99

processor	: 7
cpu		: e500mc
clock		: 1499.985000MHz
revision	: 2.0 (pvr 8023 0020)
bogomips	: 99.99

total bogomips	: 799.99
timebase	: 49999500
platform	: P4080 DS
model		: fsl,P4080DS
Memory		: 4096 MB
```

```
processor	: 0
cpu		: e500mc
clock		: 1499.985000MHz
revision	: 2.0 (pvr 8023 0020)
bogomips	: 99.99

processor	: 1
cpu		: e500mc
clock		: 1499.985000MHz
revision	: 2.0 (pvr 8023 0020)
bogomips	: 99.99

processor	: 2
cpu		: e500mc
clock		: 1499.985000MHz
revision	: 2.0 (pvr 8023 0020)
bogomips	: 99.99

processor	: 3
cpu		: e500mc
clock		: 1499.985000MHz
revision	: 2.0 (pvr 8023 0020)
bogomips	: 99.99

total bogomips	: 399.99
timebase	: 49999500
platform	: P4080 DS
model		: fsl,P4080DS
Memory		: 4096 MB
```

- On a Linux® machine, create the device tree binaries before and after changes
- Boot the Linux kernel on a p4080DS board
- Check the number of CPUs that Linux kernel sees before and after changes (use /proc/cpuinfo command)
- You should obtain the results from left side, only 4 cores are in use with the new device tree

Conclusions

- **The sequence of steps for modifying hardware device trees has been presented**

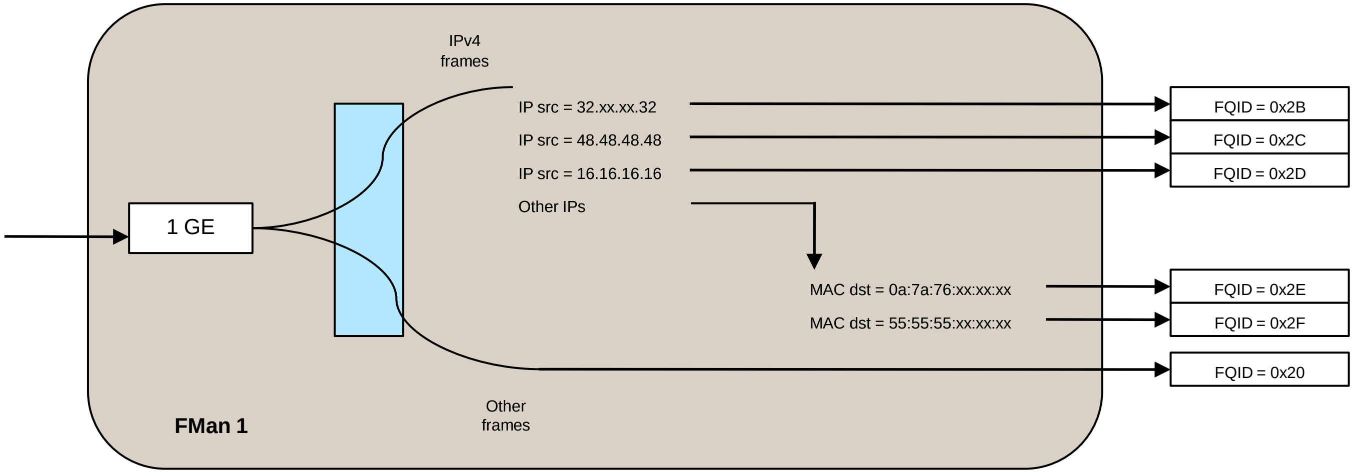
- **The benefits of using Hardware Device Tree tool are:**
  - First device tree editor including two modes for editing, GUI and text
  - Easy to understand device trees structure due to the visual representation
  - Supports device tree bindings and validation
  - Allows users to add their own device trees
  - Provides features for all the main aspects of hardware device trees
  - It is an editor and a validation tool for creating valid and well-formed device trees
  - Works on Linux and Windows hosts

DPAA Configuration

Lab 6: Using Data Path Graphs to configure the DPAA

DPAA Hands-on – Problem Statement

- **Receive 1GE traffic on first FMAN**
- **Split incoming traffic in IP frames and others**
- **Frames with specified IP source are directed in specified FQIDs and then in SW Portal 0**

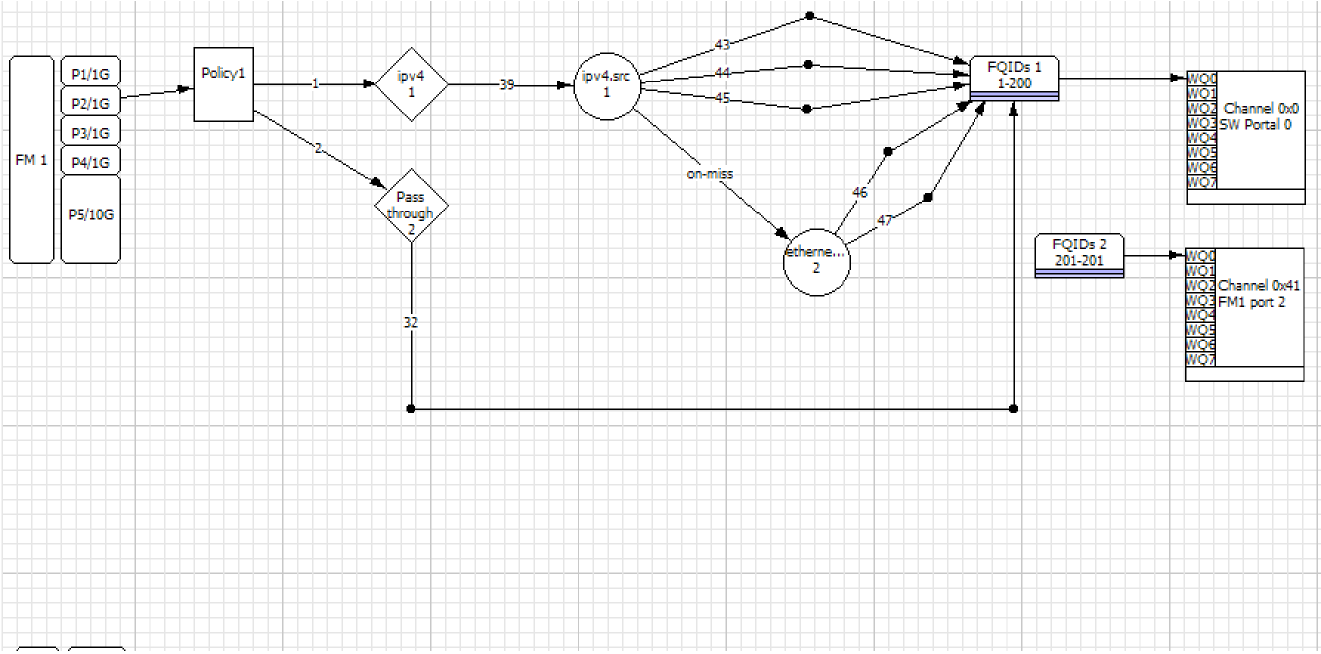


DPAA Hands-on – QCS Solution

- **Build PCD configuration according to hands-on requirements**
- **Additional 1 FQ ranges and 1 FMan port channel to be used for transmission**

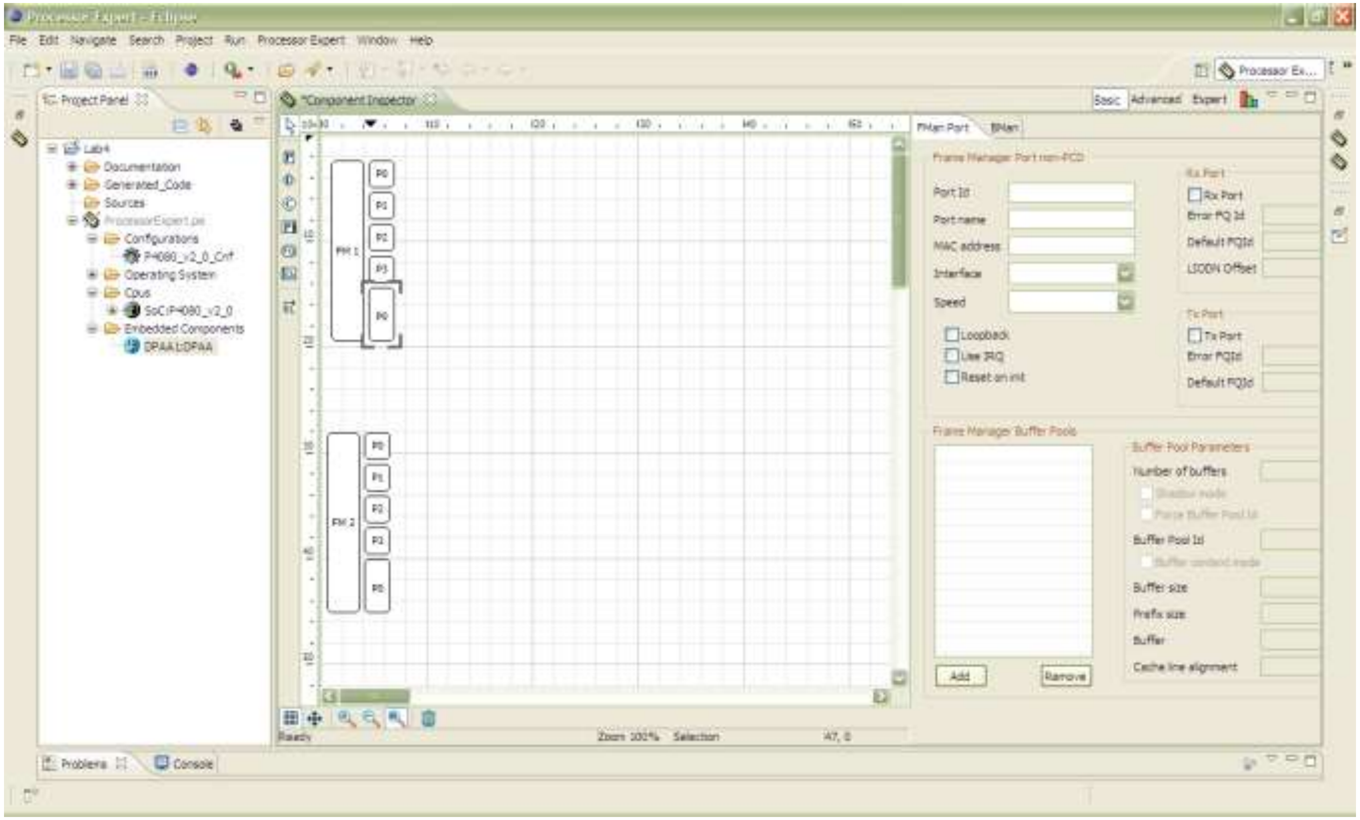
FMan1 PCD flow:

- Traffic received by FM1 port1 is split in IP frames and other frames by: Policy1, ipv4 distribution
- IP frames classified in one of the 3 defined ranges of IP source are directed into FQIDs 43-45
- All other IP frames are classified by MAC destination and are directed into FQIDs 46 & 47
- All other non IP frames are directed in FQID 32

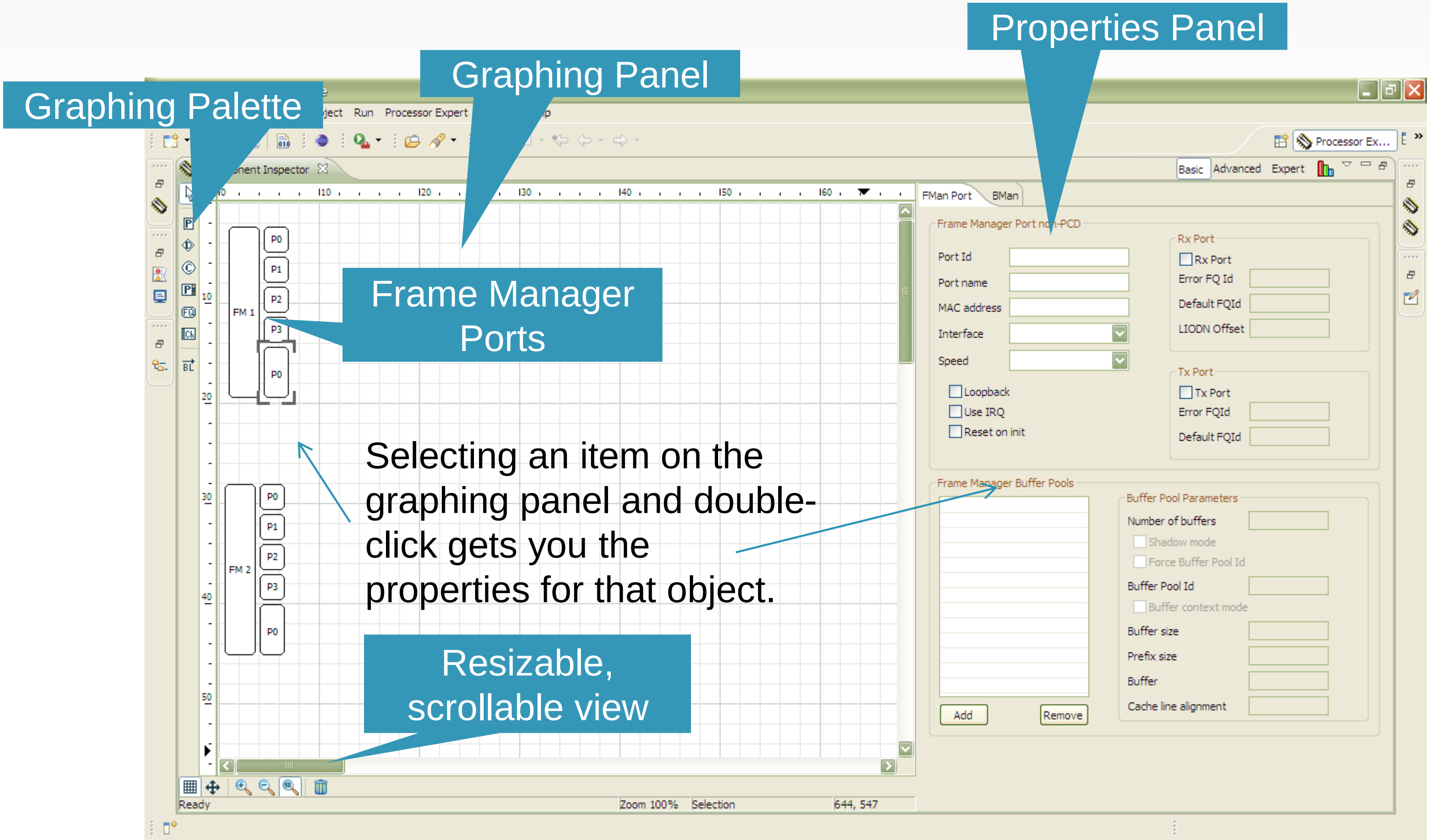


Lab 4: DPAA Hands-on

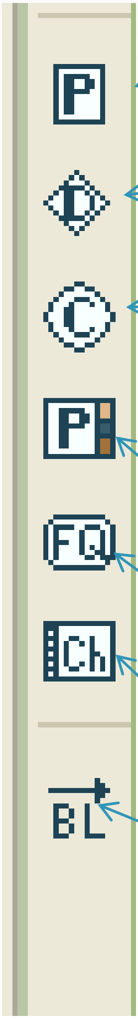
1. Create a new QCS project called Lab4
    1. Choose P4080 rev2.0
    2. Choose a DPAA component and select empty component
2. Maximize the DPAA Component Inspector



Explore the Graphing Interface



The Graphing Palette

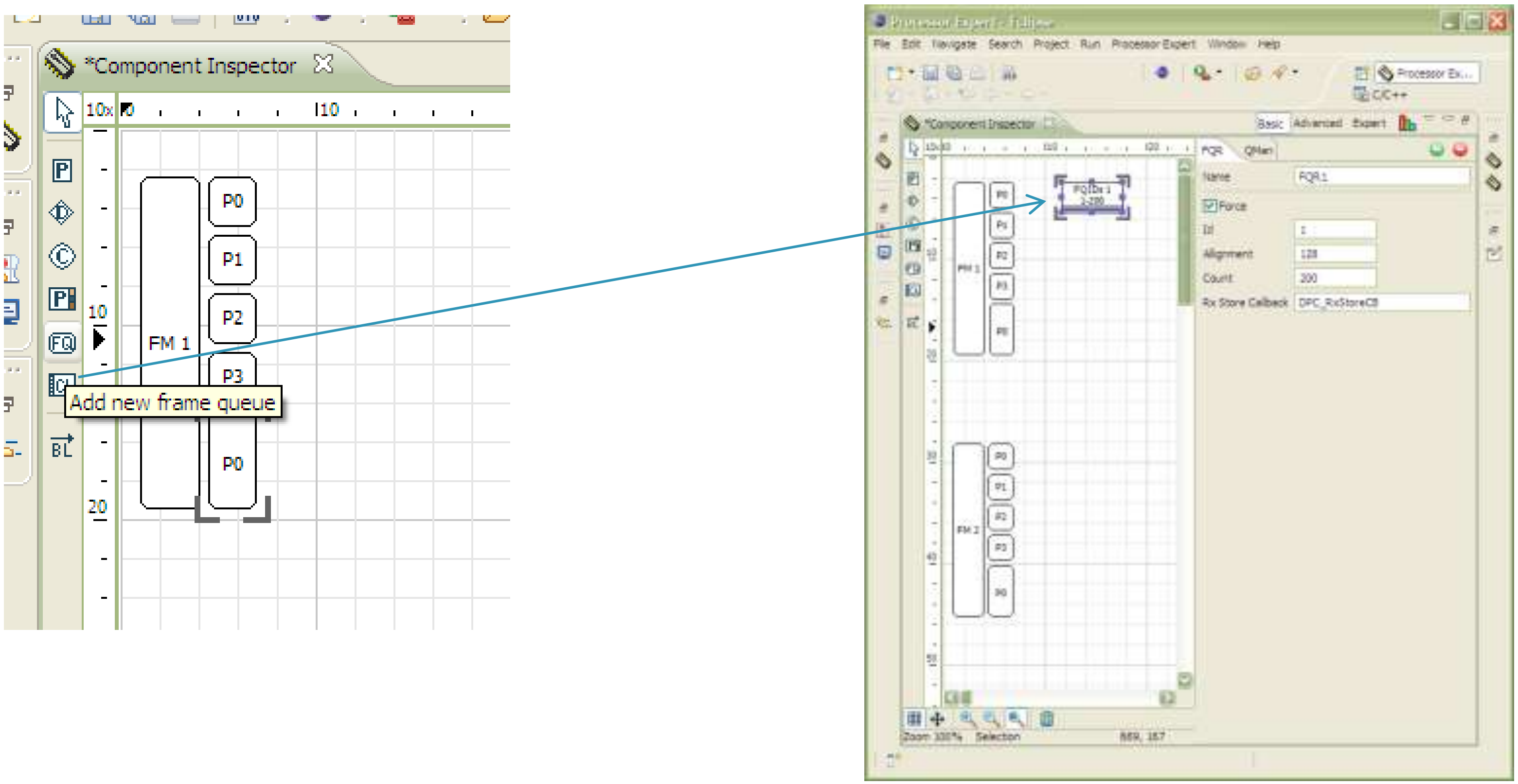


- Policy – establishes the order of distribution
- Distribution – sorts incoming frames
- Classification – defines the classification and software portal
- Policer – allows enforcement
- Frame Queue(s)
- Software Portals & Channels
- Link between objects

freescale™

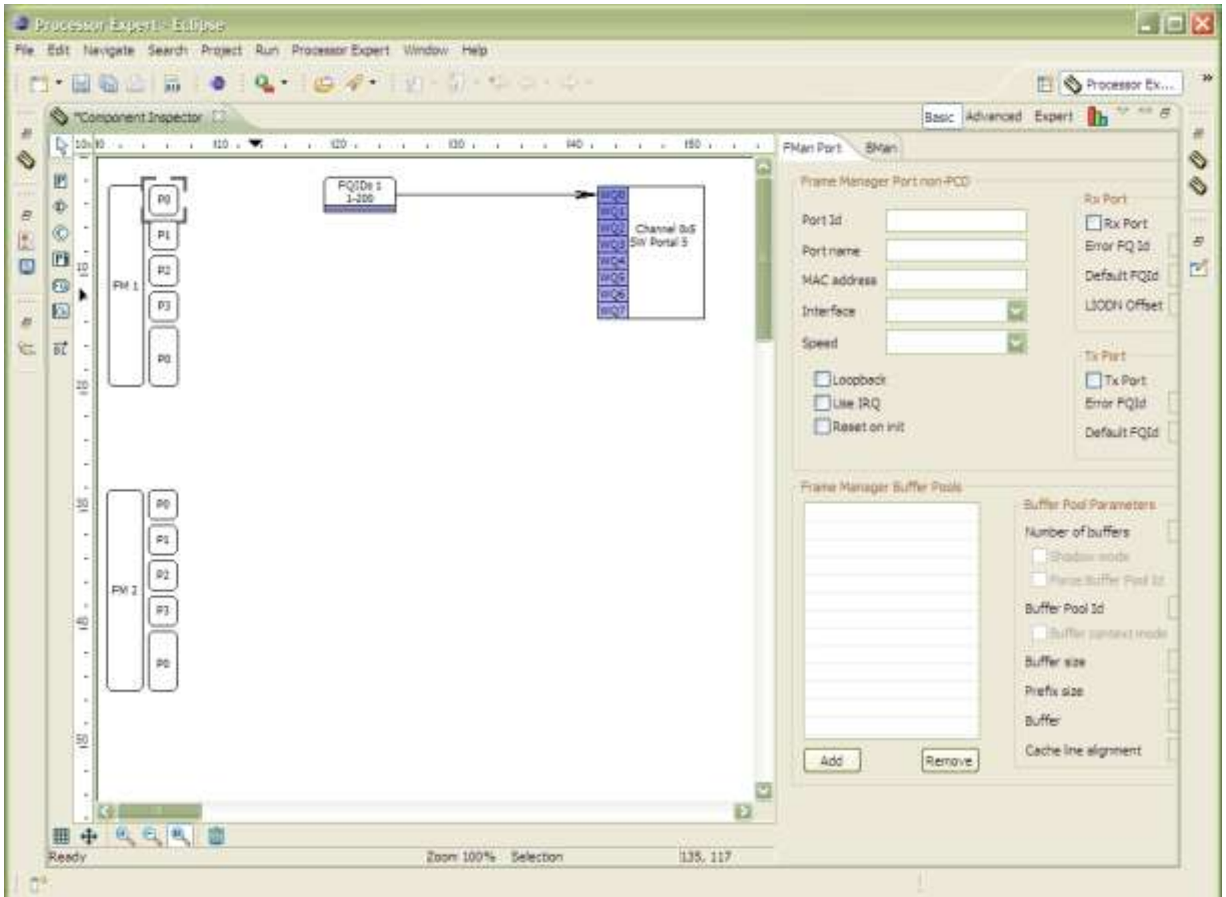
Add a Set of Frame Queues

1. QMan configuration:
   1. Add a FQIDs range (FQR1) for enqueued frames and configure: *FQId=1 and count=200*
   2. Switch to QMan tab and make the following settings: *Total FQIDs=150000, Fqd/Pfdr mem partition=Primary DDR non-cacheable*



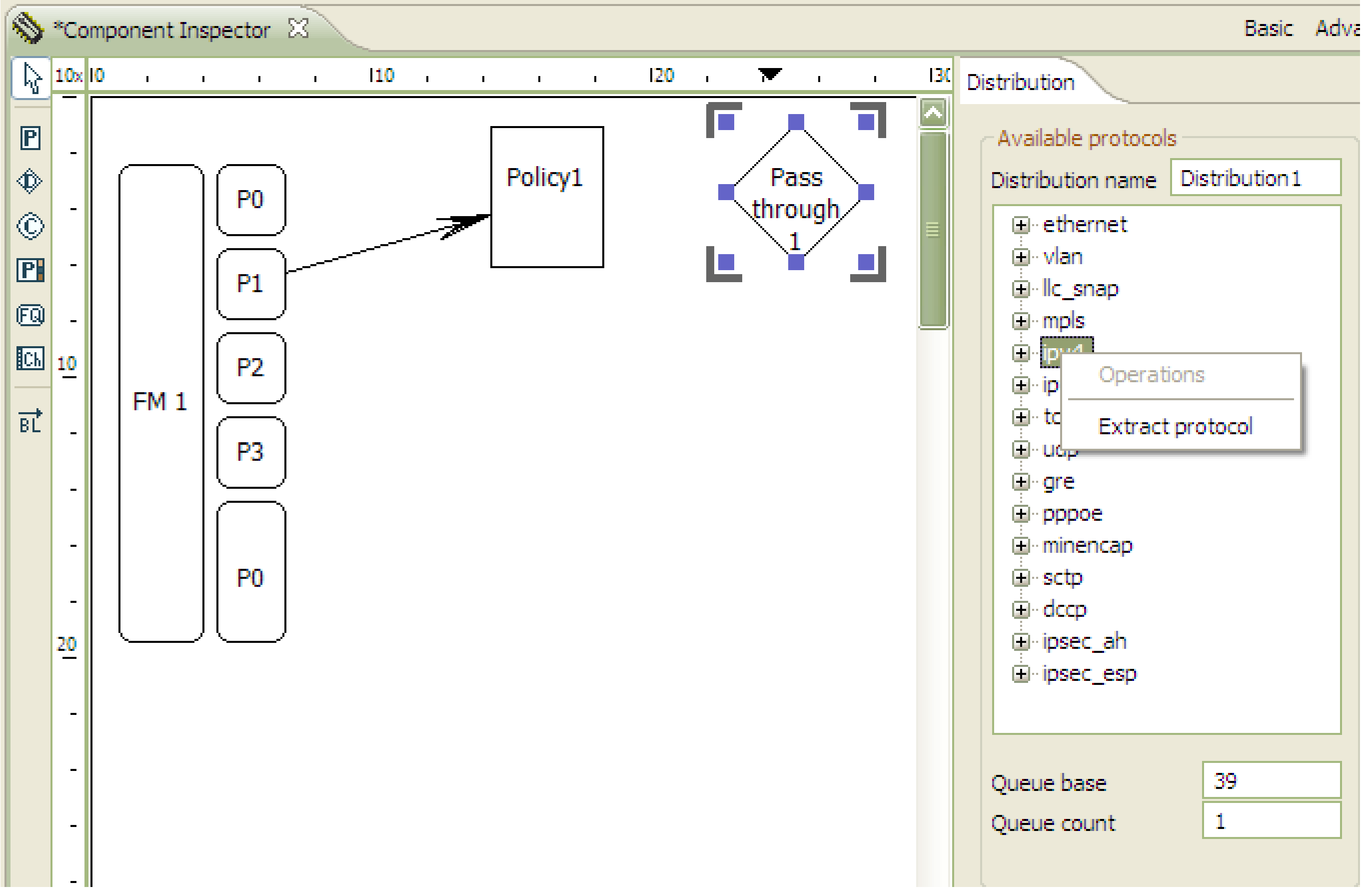
Configure a SW Portal Configuration:

1. Add SW Portal0 channel
2. Link FQIDs1 to WQ0 of SW Portal0



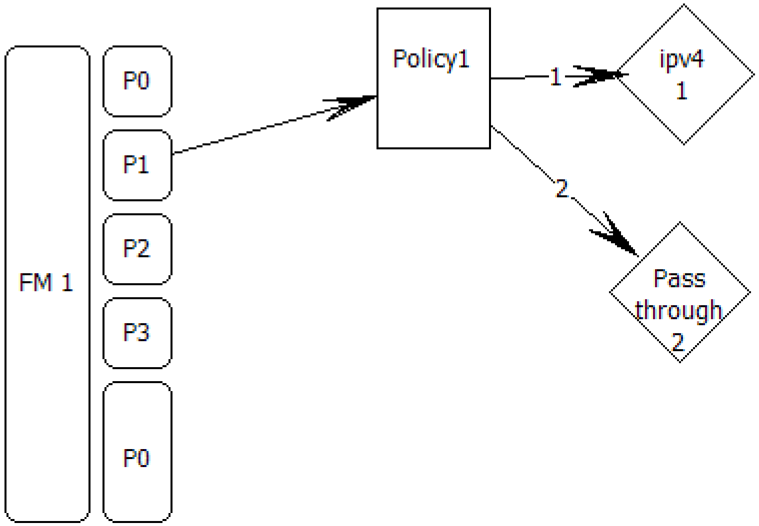
Define the Parse Classify Distribute Configuration

1. Add a Policy1 to split IP frames traffic and then a link from port FM1 P2 to receive incoming frames
2. Add a distribution for IP frames and extract IPv4 protocol then configure: *QBase = 39*



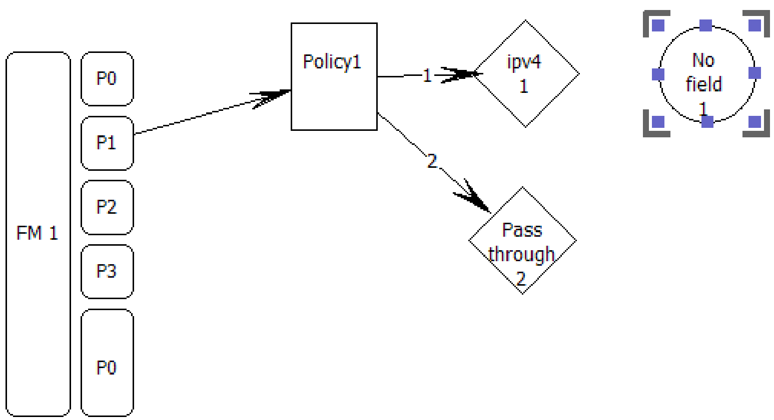
Link the Distributions to the Policy

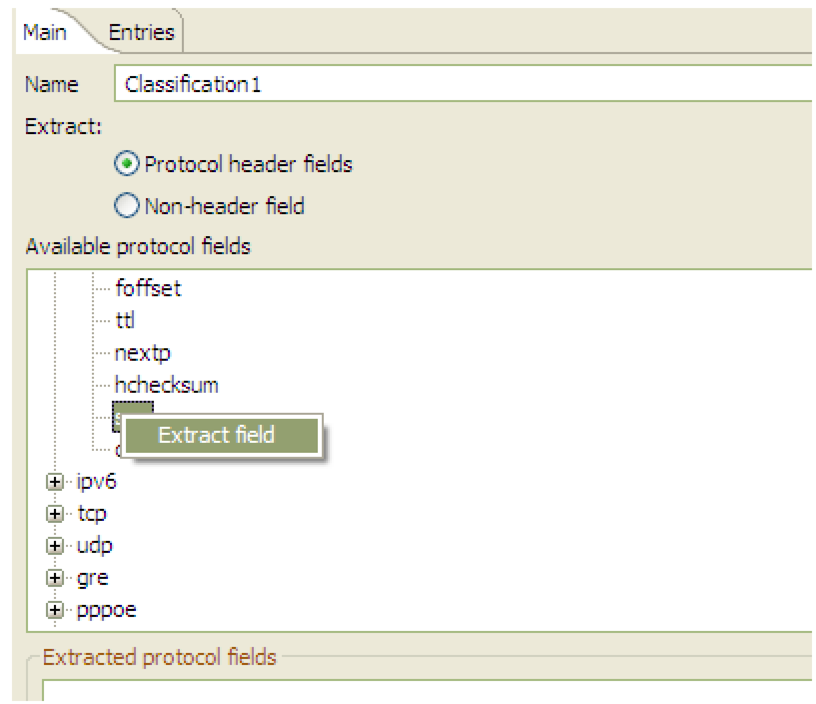
1. Add a Pass through distribution for other non IP frames and configure: *QBase = 32*
2. Link Policy1 to IPv4 distribution and then to Pass through in this order



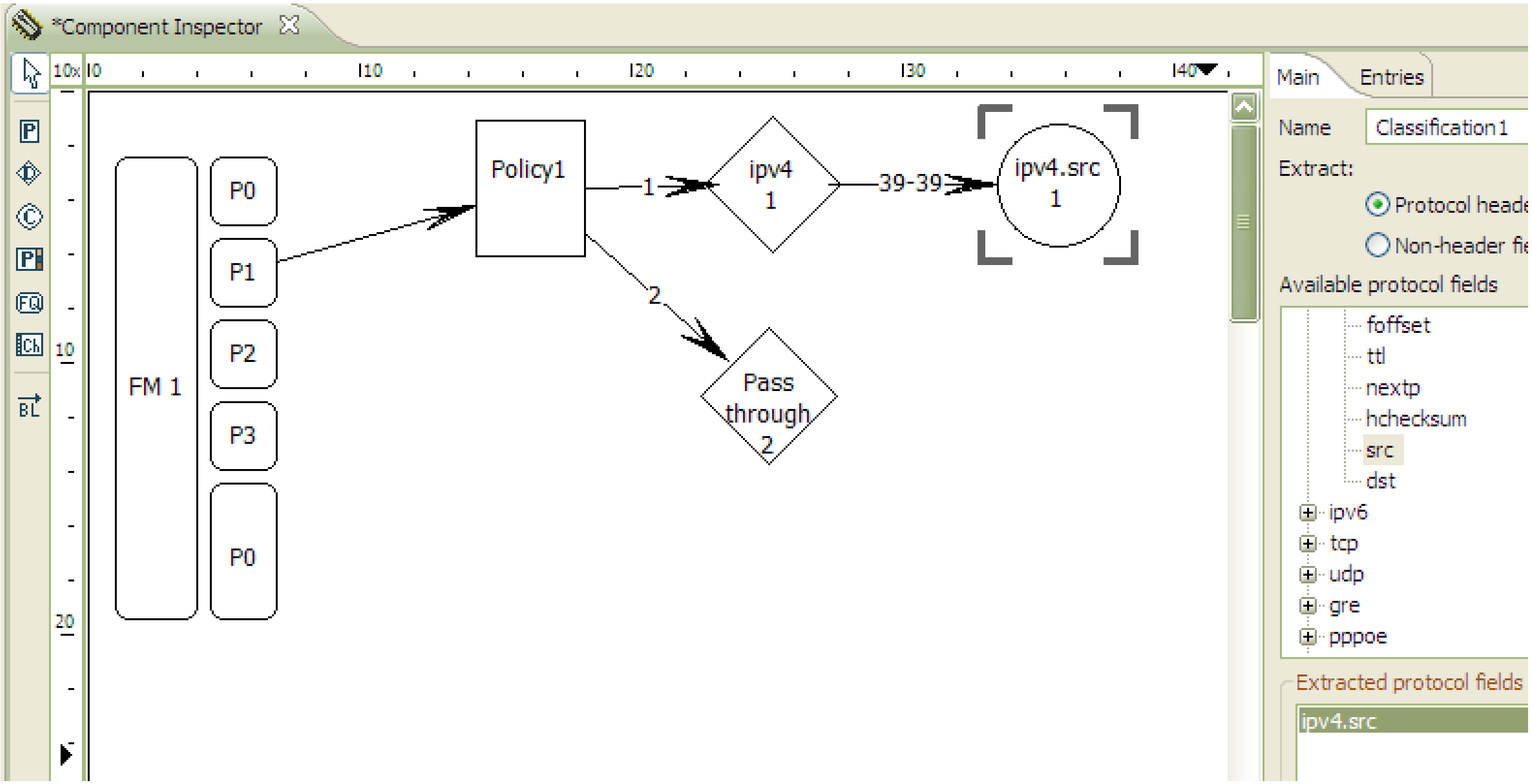
Add a Classification

- Add a classification for IP source and extract ipv4.src and remove ethernet.type



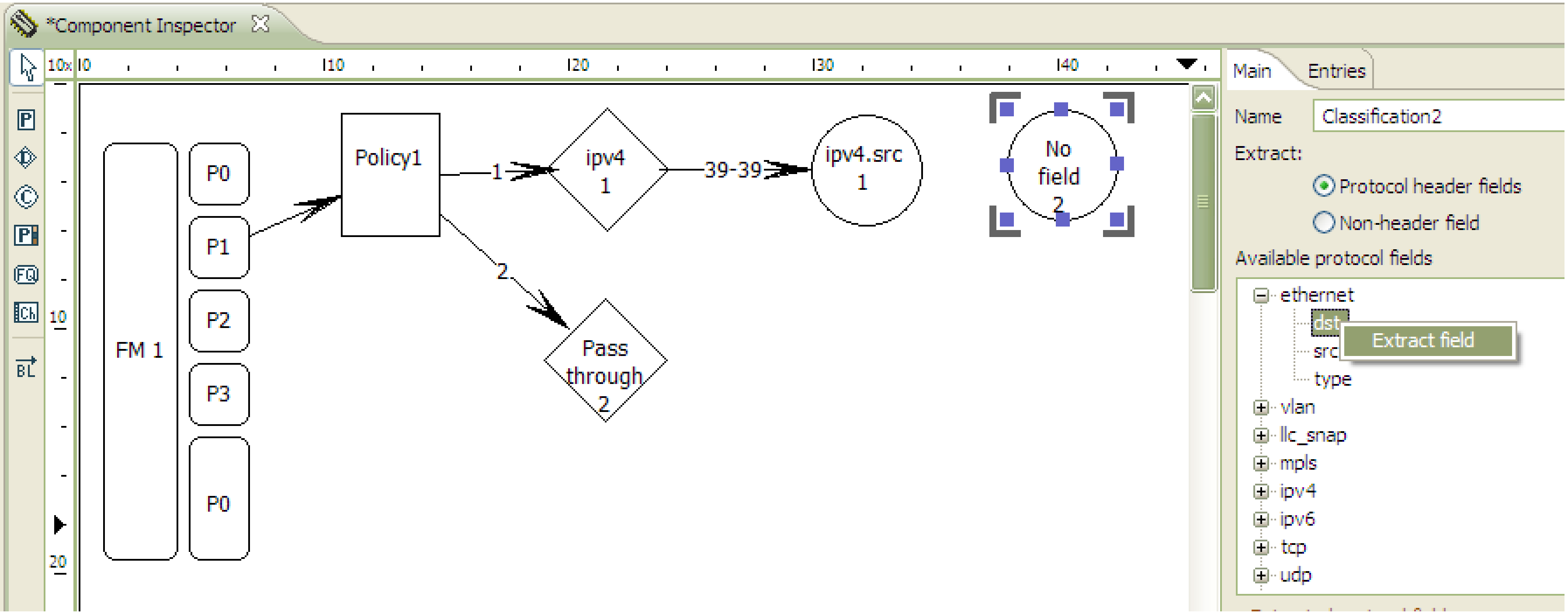


Link IPv4 Distribution to ipv4.src Classification



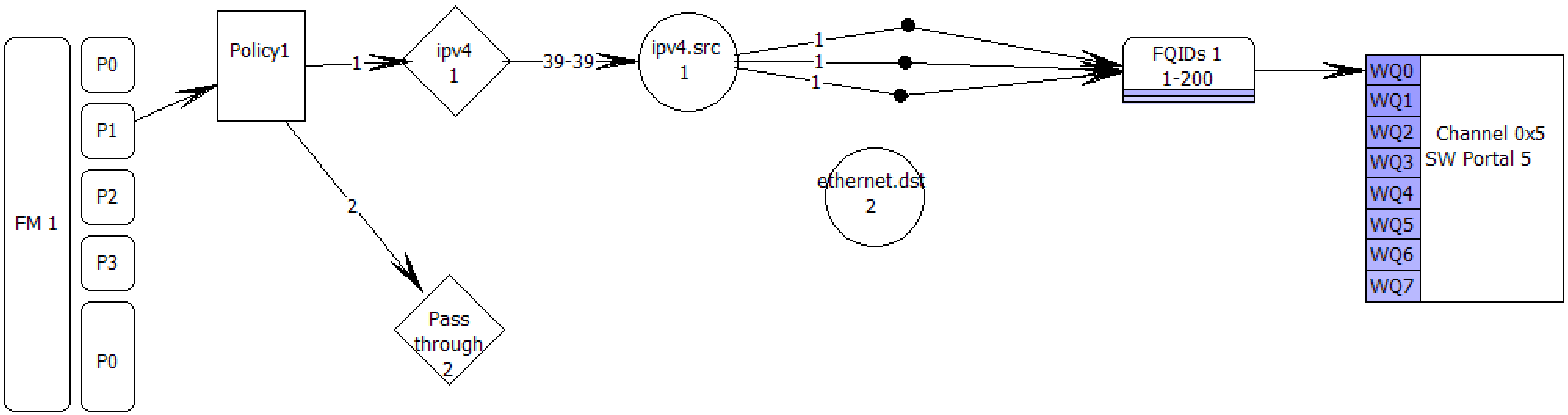
freescale™

Add a Classification and Extract ethernet.dst and remove ethernet.type



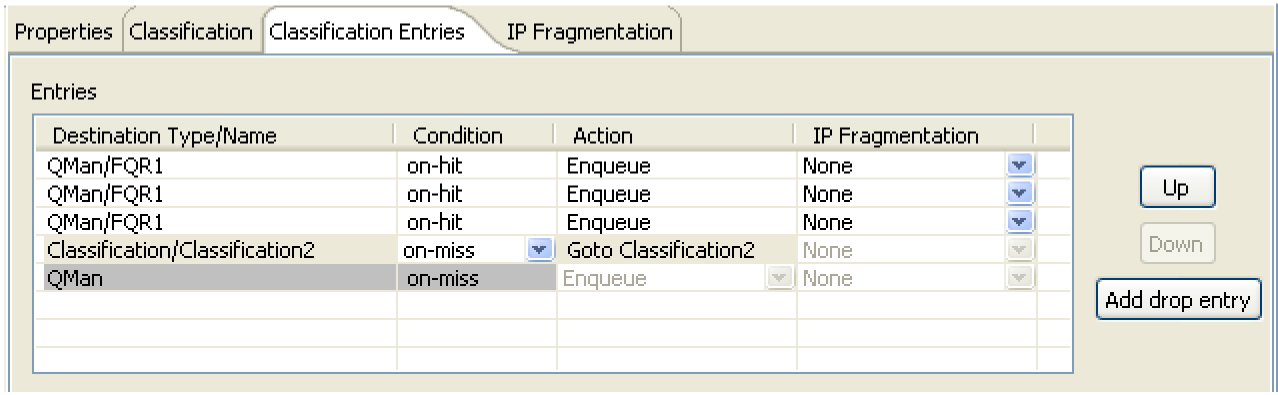
Establish 3 Classification Paths

1. Link ipv4.src classification to ethernet.dst classification
2. Draw 3 links from ipv4.src classification to FQIDs1



Configure ipv4.src Classification Entries

1. Select the ipv4.src classification
2. Go to the Entries tab in Property Panel
3. Set ethernet.dst Classification2: *on-miss entry*

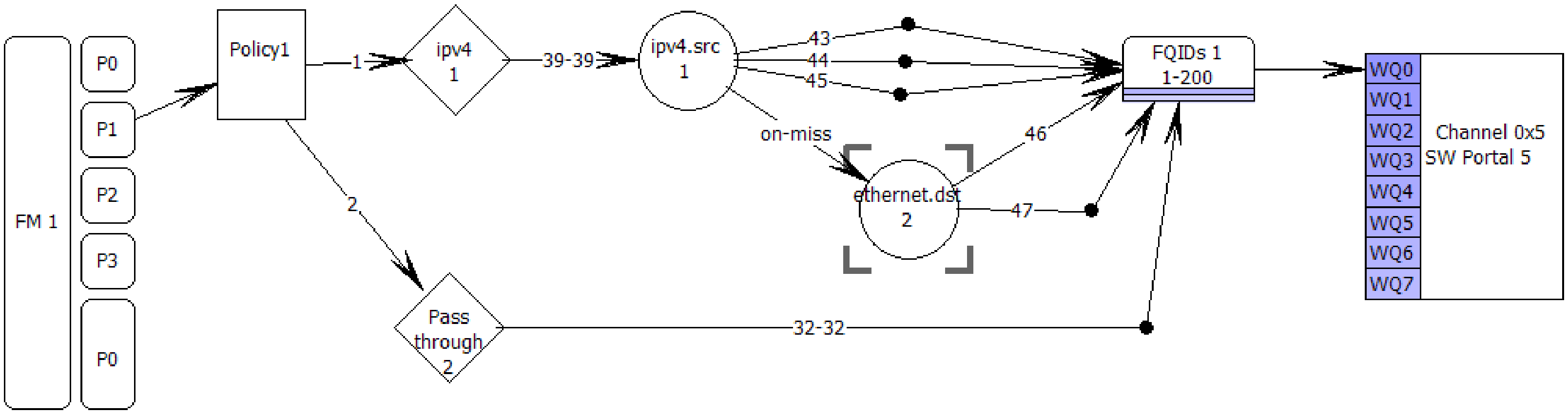


4. To FQIDs1:
   1. *Data=0x20EEEE20 / Mask=FF0000FF / Queue base=0000002B*
   2. *Data=0x30303030 / Mask=FFFFFFFF / Queue base=0000002C*
   3. *Data=0x10101010 / Mask=FFFFFFFF / Queue base=0000002D*

freescale™

Configure the ethernet.dst Entries

1. Link Pass through distribution to FQIDs1
2. Draw 2 links from ethernet.dst classification to FQIDs1
3. Configure ethernet.dst classification Entries to FQIDs1:
   1. *Data= 0a7a76000000 / Mask=FFFFFF000000 / Queue base=0000002E*
   2. *Data= 555555000000 / Mask= FFFFFF000000 / Queue base= 0000002F*



Configure the FMan Port

6. FMan 1 Port 1 configuration:
   1. *Port name= fm0port01 / MAC address=00:04:9f:00:02:66 / Interface=RGMII / Speed=1Gbps*
   2. Enable *Loopback* and *Reset on Init*
   3. Enable Rx port: *Error FQId=20 / Default FQId=20*
   4. Enable Tx port: *Error FQId=40 / Default FQId=0*
   5. Add and use BufferPool0: *Number of buffers=100 / Buffer pool Id=0 / Buffer Size=51200*
7. FMan 1 Tx configuration:
   1. Channel for FM1 Port 1:
      1. Add a FQIDs range (FQR2) used to transmit frames on FM1 and configure: *FQId=201 and count=1*
      2. Add FM1 port1 channel
      3. Link FQIDs2 to FM1 port1 channel WQ0

freescale™

DPAA Hands-on Generated Code Usage

HW configuration

- P4080 v2.0
- Loopback on FM ports used

SW configuration

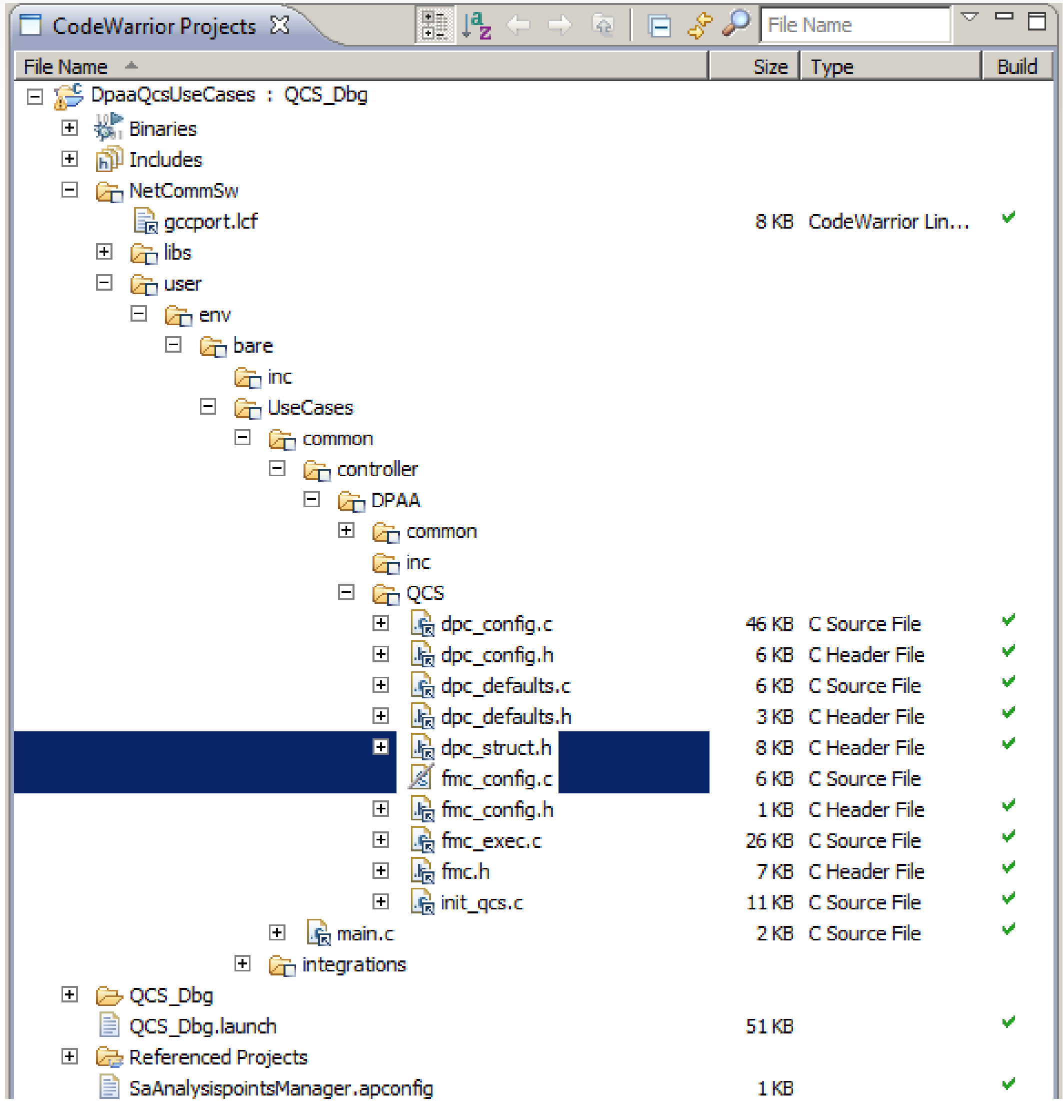
- CodeWarrior PA 10.1.2
- NetCommSw v4.5

Usecase running

- Import DpaaQcsUseCases, NetCommSw and UserEnv for P4080
- Replace generated code files:

\user\env\bare\UseCases\common \controller\DPAA\QCS

  - *fmc_config.c*
  - *dpc_struct.h*
- Clean & build project
- Run on target
- Receive output



freescale™

DPAA Hands-on Usecase Output Report

Use a serial terminal to receive usecase output

Input frames

- Frames transmitted by usecase on FMAN port2:

(5 frames on **FM0 Port**)

- IPV4
- IPV4/UDP
- VLAN
- ARP
- IPV4/TCP

Output report

- Frames received are enqueued according to PCD:

FM0 Port1:

FQID=45(0x2D)

FQID=46(0x2E)

FQID=32(0x20)

FQID=32(0x20)

FQID=46(0x2E)

```
      ####   ....
   ##########                    NetComm Device Drivers
##########                       Version 4.5
   ####                          built on May 28 2012


Usage: > <command> [options] [arg1 ... argN]
Type ? for help.
NCSW>

====> Executing Test: DPAA Basic #1:
DPAA QCS use case

> INFO (FM) [CPU00, E:/Freescale/NetComm_Software_4_5/GA_4.5/NetCommSw/Peripheral
Code]: FMan-Controller code (ver 106.2.2) loaded to IRAM.
Sending 5 frames on Port_1G_fm1_p2tx from FQR2...
Frame received on Q: 0x2d from FQR: FQR1
Frame received on Q: 0x2e from FQR: FQR1
Frame received on Q: 0x20 from FQR: FQR1
Frame received on Q: 0x20 from FQR: FQR1
Frame received on Q: 0x2e from FQR: FQR1

====> Test DPAA Basic #1 Passed !


<<< All tests passed successfully ! >>>

System is terminating... Farewell !
```

DPAA Hands-on Output Analysis

Output analysis

- **FMan0**:

Frame1: IPV4 Frame - IP src = 16.16.16.16 : IP frame -> IP miss -> MAC match -> Enqueued: FQID=45(0x2D)

Frame2: IPV4/UDP Frame - MAC dst = 0a:7a:76:5b:67:e9 : IP frame -> IP miss -> MAC match -> Enqueued: FQID=46(0x2E)

Frame3: VLAN/non-IPV4 Frame: non-IP frame -> Enqueued: FQID=32(0x20)

Frame4: ARP/non-IPV4 Frame: non-IP frame -> Enqueued: FQID=32(0x20)

Frame2: IPV4/TCP Frame - MAC dst = 0a:7a:76:5b:67:e9 : IP frame -> IP miss -> MAC match -> Enqueued: FQID=46(0x2E)

freescale™

DPAA Hands-on Conclusions

- Outcome generated code for DPAA Hands-on:
  - In order to accomplish DPAA hands-on requirements the following initialization code must be written:
    - 220 lines of XML code
    - 750 lines of C code
- Benefits of using QCS DPAA:
  - By using QCS DPAA tool the same result can be accomplished as follows:
    - Requested configuration accomplished in approximately ten minutes
    - Configuration done by using a few mouse clicks and visual parameters settings
    - Provides an easy-to-understand overview of the entire DPAA hands-on architecture
    - DPAA tool helps accomplish your desired configuration
      - by highlighting valid choices and prevent you making invalid selections
      - by performing automatic constraints checking
      - by providing instant access to configuration settings
      - by displaying relevant summary of current configuration
      - by immediate code generation at request in any stage of your work

freescale™

freescale™